Training Virtual Assistants at Scale

September 2019

Executive Summary

The models which power Virtual Assistants (VAs) are their mechanism for understanding their users. Building entry VA's is easy, creating an Enterprise Grade application that is performs consistently strong across a broad and deep corpus is challenging. Training these models is hard, and gets harder as you scale. This paper provides best practices for training and testing intent and entity models, based on experiences with multiple GBS clients in UKI. In this paper, we:

- cover the fundamental knowledge needed to work with a VA at scale;
- describe best practices for designing intents and entities from scratch, based on a combination of our own experience and expertise from computational linguistics; and
- introduce a framework and set of tools for effectively training and reliably testing VA models.

Authors

Kalyan Dutia
Consultant
Data Scientist
kalyan.dutia@ibm.com

Bella Dunnett
Consultant
Conversation Lead
annabella.dunnett@ibm.com

Contributors

Lee Taylor
Managing Consultant
Agile Test Coach
leetaylor@uk.ibm.com

Michael Conway
Associate Partner
UKI AI Practice Leader
michael.conway@uk.ibm.com

Akis Papageorgiou
Managing Consultant
UKI Chief AI Architect
konstanp@uk.ibm.com

Sebastian Weir
Senior Managing Consultant
Cognitive Strategist
sebastian.weir@uk.ibm.com

Table of Contents

Why Training at Scale **3**

Overview **4**

Skill Architecture 4

Intents and Entities: Under the Hood 6

Measuring Performance 8

Training Data Selection: Best Practice **11**

Constructing Intents 11

Deciding Utterances 14

Creating Entities 16

Iterating & Improving **18**

The Role of the Model 18

Exploratory Analysis: Positioning the Starting Block 19

Improving the Model: Targeted Training Approach 22

Conclusion **27**

Why Training at Scale

The key to any good conversation is comprehension. There is nothing more frustrating than trying to explain an urgent issue to either a person or virtual assistant who does not have the comprehension to understand, or the training to respond. Conversely, the best customer experiences stem from meaningful, practical dialogs. If a virtual assistant is to provide a meaningful and sustainable augmentation of a customer engagement channel then the growing capability is critical to its success, but this growth must be built on a stable, measurable and transparent platform. As the capability To understand, advise and assist grows, it is vital that its training data enables the correct classification of a user's query.

The processes behind Training at Scale provides recommendations for the best use of available utterances data in order to create a more positive and helpful user experience. The structure of a skill[1], and the training data that comprises it, are essential components of a successful virtual assistant. Training at Scale also aims to tackle some problems we've seen for large-scale VAs, including:

- Ensuring that the VA continues to understand its users, as the language they use may change
- Minimising confusion between existing and new intents, when new intents are added at a fast rate
- Managing the ever-increasing quantity of training data over a large number of intents
- Running essential corpus tests which provide results reflective of the current user experience
- Capturing utterances which have been considered 'out of scope' in order to expand the range of queries to which the VA can respond

Though the principles in this paper have been designed for virtual assistants at scale, they are just as applicable to VAs with fewer intents and fewer chats per day. Regardless of a virtual assistant's size, the priority should always be the quality of its classification. If a VA's classification is not functioning as it should, then it will not understand its users and consequently will fail to provide a correct and helpful treatment – fundamentally this will

[1] Watson have recently renamed workspaces to skills. To avoid ambiguity, we'll stick to using the new 'skills' terminology in this paper. If you look at the referenced REN paper, you'll see the same architecture described using 'workspace'.

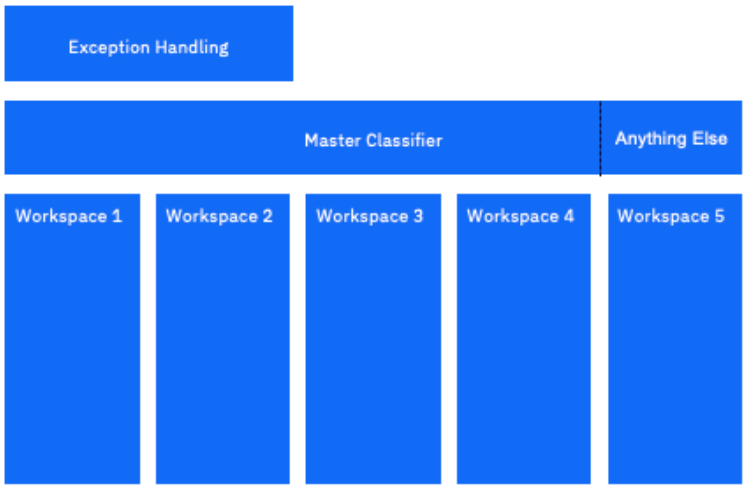
create a barrier to the channel for customers and never provide the Business value envisaged

Overview

In this section we describe what we consider to be the fundamental knowledge required to build an effective VA; namely structuring skills according to the REN architecture[2], understanding how a Natural Language classifier such as the Watson Assistant, and measuring the performance of this classifier.

Skill Architecture

The training data contained within a virtual assistant will be dependent on its scope and intended coverage. This is not to say that the content or coverage of the VA will differ substantially, but rather that the intents must be structured in a different way so as to distinguish them from similar intents within the same skill. Different processes must be followed depending on whether a VA uses multi-skill architecture or single-skill architecture. When chosing the appropriate architectural approach is it essential to consider the VA's intended audience to ensure you anticipate the potential growth a overall Corpus.



Multi-Skill Architecture

The REN architecture utilises a master skill and multiple domain-specific skills to triage user requests. This architecture functions successfully when building a VA at scale, enabling utterances to be assigned correctly to a lower, topic-based ('domain') skill. The VA is able to distinguish more closely between intents in each given topic skill, and so respond to a wider number of queries and provide a more detailed response. The image on the right represents the skill layout of this architecture.

When using different topic skills as per the REN architecture, the training data within each skill must be separated by actions rather than topics. For example, if you have a topic skill related to mobile phones, the intents within this skill should relate to the actions that the

[2] See more details in the Conversational AI UX Guide: https://w3-03.ibm.com/services/lighthouse/documents/128090

user will want to perform related to mobile phones. Thus, 'I want to buy a new phone' would equate to #PurchasePhone, whereas 'My phone is broken' would be defined as #BrokenPhone.

The variation in the training data in these two intents should centre around different verbs, rather than different nouns. The focus should not be on varying the word 'phone', as the word and its synonyms will appear consistently throughout the skill. Rather, each intent ought to be distinguished by the specific action that it describes. Therefore, to boost classification performance on a multi-skill architecture, intents should be comprised of numerous relevant verbs. To revert back to the earlier example of #BrokenPhone, other utterances might include 'I think I've damaged my phone', 'Phone has been smashed', or 'Large crack on my phone screen'. Each utterance refers to the topic 'phone', but varies in terms of sentence length, structure, and action synonym. It's worth bearing this in mind before tackling the construction of your intents.

Single-Skill Architecture

For smaller use-cases with a less extensive knowledge-base, it is possible to implement a single-skill architecture. Unlike multi-skill architecture, where the topic has already been defined, training data can be collected that spans multiple topics and multiple actions.

However, a mix between action driven (#Purchase) and topic driven (#Insurance) intents can lead to complications with classification. Let us imagine that #Purchase relates only to queries relating to purchasing items, and that these items have been distinguished through entities in the dialog flow. #Insurance, meanwhile, will cover all queries relating to insurance - and crucially also includes queries relating to *purchasing* insurance. These utterances in the training data that relate to purchasing insurance will be structured very similarly to those relating to purchasing a new phone. It would be difficult for the classifier to distinguish between these two intents and could easily lead to users receiving the incorrect treatment for their query.

Though there will always be cases where intents must be structured based on topics, it's valuable to consider what actions exist within these intents before assigning training data. If the topic intent is little more than an entity value that could be added into other pre-existing intents, then it should not be created as an intent in its own right. Whilst this might lead to more complex dialog flows, the benefits of having a better-performing classifier will far outweigh this inconvenience.

This issue, though more applicable to single-skill architecture, is also relevant to VAs using multi-skill architecture. Unless there is a comprehensive awareness of all training data within intents, and what those intents ought to contain, there will always be the capacity for clashes and overlaps within skills.

Intents and Entities: Under the Hood

A VA understands utterances through intent and entity classification, and good classification performance ultimately means that you can design a dialog flow knowing that utterances are more likely to trigger the nodes that you expect. In order to improve and measure classification, it's useful to have a high-level understanding of how intent and entity classification work under the hood.

Intent classification works using machine learning. This means that understanding is not reliant on exact key word matches and the VA has an awareness of the semantic similarity between words, as well as their relative place in an utterance.

A general guide to the Natural Language Processing (NLP) that occurs in classifiers such as Watson Assistant's (WA's) follows.

Text Pre-Processing

Some basic transformations are applied to all utterances, in order to improve the stability of classification.

A. Convert the utterance to lowercase. This means that WA is not case sensitive.

B. Convert any common emojis to their text equivalents, e.g. 😊 to :-)

Feature Extraction

Several 'features' are used to represent each utterance to the intent classifier. These are compiled into a 'feature vector' representing the utterance, which could have a 1 if the feature is present, and 0 if it is not.

- **Ngrams.** An ngram is defined as a series of consecutive n words in an utterance, e.g. "new phone" would be a 2-gram (bigram) in the phrase "I would like to buy a new phone". This means that the classifier has some information about the local position of each word within an utterance. A process called *stemming* is performed on each word, to reduce it to its root (e.g. buying -> buy; phones -> phone). Both forms of the word are used in classification.

- **Word embeddings.** These are vector representations of words which have been learnt by a neural network, in which semantically similar words have vectors which are closer together. This means that the intent classifier has some understanding, for example, that the words *woman* and *man* have a similar relationship to *sister* and *brother* respectively.

Classification

The classifier uses all the information available in the features of each training utterance to build a model which tries to predict the best matching intent for an utterance.

Part of this process involves applying a 'weight' to each feature in the intent, where features with a greater weight will affect classification more than those with a smaller weight. You can expect common words such as 'a', 'an', 'could', and 'may' to have lower weights, whereas rarer, more topic-specific words such as 'receipt', 'contract', and 'SIM' will have higher weights.

For example, in the utterance "I would like to buy a new phone", the words 'new' and 'phone' will have much more influence on the predicted intent than 'I' and 'would'. However, repetition of 'I' and 'would' in a particular intent will still bias the classifier towards that intent - thus repetition should be avoided if this is not desired behaviour.

It is also worth noting that WA's intent classifier is tailored for shorter utterances (< 1024 characters). If you are involved in an application which involves classifying longer pieces of text, it will be useful to compare the performance of Watson Natural Language Classifier against the WA intent classifier.

Entity detection is a lot simpler than intent classification on the surface. With fuzzy matching turned off (recommended[3]), WA looks for an exact match between each word in an utterance, and the list of entity values and their synonyms.

However, relying on exact matches means that you can no longer take advantage of the understanding of similar words (word embeddings) and roots of words (stemming) that exists in the intent classifier. When designing entities, this means that you should take care to include an exhaustive list of synonyms, as well as plurals of all words. The synonym recommender feature in WA is useful for this.

Resting entities make use of a side-effect of adding words as entity values, even if they aren't included in the conditions of any dialog node. Including a word as an entity value or synonym artificially increases its 'weight' in classification, with the increase being greater for an entity value than a synonym.

You may want to exploit this behaviour to improve the performance of a particular intent. For example, an intent called #Insurance that covers all queries relating to insurance would be boosted by an entity called @Insurance that includes various different synonyms of the word.

[3] Fuzzy matching enables entity matching for misspellings or different grammatical forms of words. In practice we've found that this leads to a lot of unwanted matches to entities, e.g. 'cancel' triggering the entity value 'cancer'.

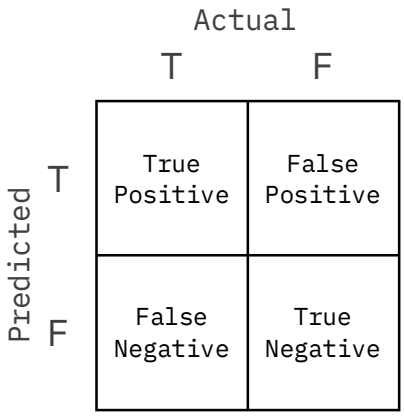
Details and examples on how to use this information to design effective intents and entities are in Training Data Selection: Best Practice.

Measuring Performance

In order to make reliable changes to training, it's vital that you have a way to measure its performance. This section outlines how to choose the best metrics to use, and how to ensure that you have a reliable, stable test process.

Basic Metrics & Confusion Matrix

It's important to choose metrics that represent ideal user experience, given the architecture of your VA. That way if you aim to maximise your metrics, you'll also be working to optimise the user experience.



Sample confusion matrix

The fundamental building blocks for corpus metrics can be read from a diagram called a confusion matrix, which can be seen on the right (we'll later expand on how to see a similar diagram to identify misclassifications). The four quadrants of the confusion matrix relate to whether the VA has served a response (the example is 'positive' or 'negative') and whether it is correct (the example is 'true' or 'false').

Whilst changing training data is the main way to affect the balance of the confusion matrix, you can also change the ratio of positives to negatives using a *confidence threshold* as a condition in all dialog nodes. The dialog should be designed so that the user is only served a response if the intent in question is triggered with confidence above this threshold.

When looking at the performance of an intent corpus, the four quadrants of the confusion matrix have definitions as follows.

Metric	***Definition***	***Example Utterance, Expected Intent***	***Example Classification (0.4 threshold[4])***
True Positive (TP)	Confidence ≥ threshold, Correct first intent	"Can you tell me about phone insurance?" #Insurance	#Insurance, 0.65

[4] The confidence threshold tends to be set in WA, above which the user will be served a response, and below which they will not. Methods for finding an optimal threshold are touched on later in this paper.

False Positive (FP)	Confidence ≥ threshold, Incorrect first intent	"Can you give me a quote for a screen replacement" #BrokenPhone	#ContractIssues, 0.89
True Negative (TN)	Confidence < threshold, Incorrect first intent	"I'm coming to the end of my contract soon, how much is a new one?" #ContractEnd	#ContractIssues, 0.24
False Negative (FN)	Confidence < threshold, Correct first intent	"I want to get a new phone" #PurchasePhone	#PurchasePhone, 0.20

If you are heavily using entities in dialog nodes in conjunction with intents, measuring intent performance alone will not give you a full picture of the quality of user experience in the VA: there may be instances in which the intent performs as designed, but the dialog node is not triggered because of a misspelling or a missed synonym within an entity. Therefore, in addition to intent tests you may want to consider using a confusion matrix centred around the responses that the VA serves when tested with a series of utterances. Here, positive/negative will correspond to whether the VA has served a response, and true/false will correspond to whether this was the correct action based on the utterance.

Developing Informative Metrics

The next step is to find metrics which reflect the ideal customer experience. Some common choices are

- **Precision = TP / (TP + FP)**
  The proportion of messages for which the VA serves the correct response, out of all the times it serves a response. A low precision intent could be seen as 'greedy' - it is matching lots when it shouldn't be.

- **Recall = TP / (TP + FN)**
  The proportion of messages for which the VA serves a response, out of all the times that it correctly identified the intent. Low recall could be seen as a 'shy' intent - it often doesn't give an answer, even when it's correct.

- **Accuracy = (TP + TN) / all**
  The proportion of times the intent model does the right thing: either serves the correct response or doesn't serve a response when it's got the intent wrong.

- **F_1 score = 2 (P x R) / (P + R)**
  The *harmonic average* of precision and recall. If you care about one of precision or

recall more than the other, you may want to consider an F_β score, which allows you to specify exactly how much you care.

In order to choose the best metric to measure your VA, it's best to go back to the confusion matrix, and think of the user experience when a user encounters a classification in each of the four quadrants. For example, in skills with clarification[5] a False Negative is less costly than a False Positive, as the customer still gets to direct the conversation by clicking on a button - therefore it's better to focus on improving precision rather than recall.

It's important to note that confidence thresholds have an impact on all of these metrics, as they do on the user experience. This can work in your favour: there are some tools available[6] which help you to choose the confidence threshold which maximises your chosen metric(s), a process that could be run every release.

Running Reliable Tests

The final piece of the puzzle when measuring classifier performance is having tests which you can use to measure your chosen metrics. Ideally you want to be able to use a 'test set' made up of utterances which are representative of what users are saying to the VA in production, although there may be limitations, such as data access, which could prevent this.

There are 3 main types of corpus tests available, each with their own advantages and limitations:

- **Blind set testing.** A selection of utterances which weren't used in training is pulled from production logs and labelled with the expected intent (you could also label the utterances with expected entity values or responses, and check for a match). These are then fired at the VA using a testing tool[7], which checks whether each intent has matched the expected intent, and its confidence. Each utterance and its response can then be categorised according to the confusion matrix, and metrics calculated from these.

  + Best represents how the VA behaves in production.
  - Work is needed to continuously update blind sets, and check that they're representative of production chats. This can be time consuming.

[5] A mechanism for prompting a user to select their query from a list of buttons, if a topic is detected but with too low a confidence to serve a response. See the Conversational AI UX guide for more details.
[6] https://github.com/IBM-DSE/Watson-Assistant-Testing-Tools
[7] We are currently developing a testing asset in UKI, which has a UI and is deployable to IBM Cloud. Please reach out to the authors for more details.

- **K-Fold testing.** This is an example of a 'cross validation' method which doesn't require labelling of external data. The training data for a skill is split into K (usually 5-10) folds, and then one is used as a temporary 'test set' and fired against a temporary training set comprising utterances from the remaining K-1 folds. This process is then repeated for each fold.

  + Just requires your training data.
  - Never tests the full training set as it exists in production.

- **Monte-Carlo testing.** A process very similar to K-Fold, where each iteration's temporary test set is decided by randomly sampling a percentage of the skill training.

  + Similar to K-fold, but quicker.
  - Can't guarantee that all your data has been tested due to random sampling.

In a VA running at significant scale it's common practice to have both **validation** and **QA** processes. The validation process is owned by the team of conversation builders and allows them to use the aforementioned tests to validate that changes they have made show an improvement in the VA's performance. The test process is owned by the QA team (although the conversation builders may need to help create the test set), and used to ensure quality for each release, similar to general software development practices.

The table below shows recommendations for each type of test, for validation and QA, split into high- and low-volume VAs. We define a high-volume VA as one where there is enough production data to create reliable test sets, with at least 20-25% as many utterances as your training set.

	Low-Volume VA	**High-Volume VA**
Validation	K-fold	Validation Set
QA	Monte-Carlo or Blind Set	Blind Set

Recommended test approaches for high- and low-volume VAs

Training Data Selection: Best Practice

Constructing Intents

A common challenge when building a virtual assistant is to convert a client's data into an effective series of intents. A list of FAQs and example utterances does not necessarily equate to an equal list of intents. Indeed, it is often better to ignore the FAQ 'names' altogether and look only at the utterances that they contain. An FAQ titled 'Check

Paperwork Sent' might be comprised of very similar utterances to 'Paperwork Arrival Time', though they have been classed as separate questions. In both cases a user utterance might look something like: 'Why haven't I received my paperwork yet?' or 'It's been two days and my letter still hasn't come'. These FAQs could be merged into a single treatment, where the user is presented with the typical shipping time and then offered a handoff if they confirm they have been waiting for longer than this time period. When defining this list it is recommended to work close with Business SME's and call centre agents, as they have first hand knowledge of how the Virtual Assistants future customers converse.

In order to classify new training data correctly, it's recommended to export all utterances into a single spreadsheet and label intents and entities individually. The columns should read *utterance*, *main topic*, *focus 1*, *focus 2*, and *focus 3*.

Utterance	***Main Topic***	***Focus 1***	***Focus 2***
Why haven't I received my paperwork yet?	Delivery	Paperwork	Issue
It's been two days and my letter still hasn't come	Delivery	Letter	Issue
I'm worried that my paperwork has been posted to the wrong address	Delivery	Paperwork	Wrong Address
When will I be emailed that letter?	Delivery	Letter	

The table above indicates how various utterances could be broken down into intents and entities. The words 'received, 'come', 'posted', and 'emailed' all relate to issues surrounding delivery, and so can be considered under the same umbrella term. In this instance, it does not matter that 'posted' refers to a physical document, whilst 'emailed' likely refers to a PDF. Regardless of semantics, the action behind the intent centres around when an item is likely to arrive.

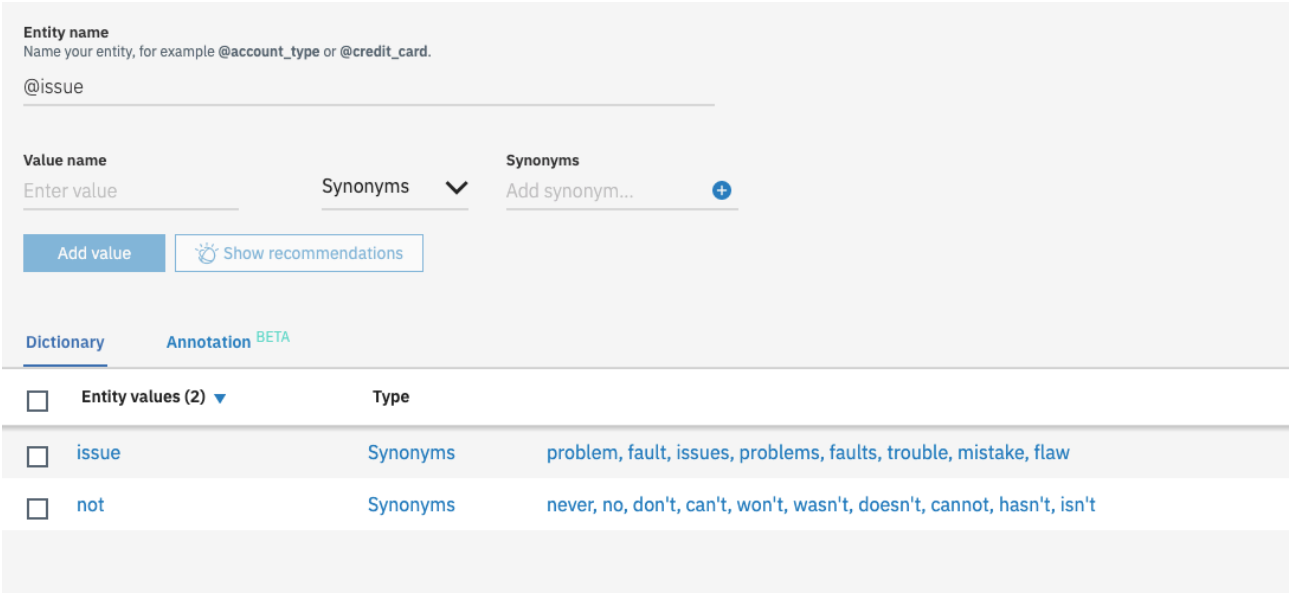
The next step of analysing these utterances is to determine what the focus is of the statement. It is important to think about which details differentiate one statement from another. In this case, the main topics are 'paperwork' and 'letter', though these are not the only entities that you might want to create.

Consider the two example utterances 'It's been two days and my letter still hasn't come', and 'When will I be emailed that letter'. Both dialog flows would match the intent

#Delivery and then the entity @letter, but there is more to these statements than this. The first statement is relating to a perceived issue with the delivery of their letter, whilst the latter is merely querying the expected time-frame. This slight difference means that both users require a slightly different treatment: the first will likely need to speak to an agent, whilst the other user will need an FAQ style response.

In order to achieve this distinction, an entity should be created to establish that an issue is present. This entity, @issue, ought to contain all negative forms of verbs, and various words that denote a problem. It is not recommended to create two intents for this purpose - #Delivery and #DeliveryIssue - as the utterances will look very similar and consequently confuse the model.

Your entity might look something like below.



The list of entity values and synonyms seen above is not exhaustive and will contain numerous words specific to any given skill. For example, the 'Phone' skill mentioned earlier would have a specific entity value for 'broken', with a number of relevant synonyms. These entity values should only be seen as a guideline for your own skill.

As you have seen in this example it's possible to extract more from an utterance than just the intent, using entity matching to pick up extra details from an utterance that intent classification would miss. Though this process is more manual than pure conversion of FAQs to intents, it will lead to the creation of a much more effective set of intents for your virtual assistant.

Deciding Utterances

Once you have determined a list of intents, it is time to start thinking about the utterances you will use to train your VA.

At this stage, your initial corpus of training data ought to have been sorted according to your defined intents. It is now necessary to reduce these utterances further in order to distil them into the best possible training data for your model.

Generally speaking, the fewer utterances you use the better. Anywhere between ten and twenty utterances for a simple intent is ideal, though this number is likely to increase if an intent has more variation in its verb form. For example, an intent called #LostPhone might contain only a few verbs, such as 'lost', 'mislaid' and 'misplaced'. As such, it will likely need a lower quantity of utterances than more diffuse intents. By contrast, an intent called #ApplyInsurance might contain verbs 'get', 'apply', 'arrange', and 'have'. You would expect this intent to need many more utterances in order to train the model on all variations. Intents such as #ApplyInsurance might need as many as 60 utterances, though any more than 60 is not recommended and anywhere above 100 may mean that you are introducing unnecessary redundancy, which will not improve performance and will be more difficult to manage.

Aside from their volume, the content of these utterances is also vital to intent classification. The following considerations ought to be taken into account when selecting utterances for training data:

- **Keep a ratio of about 9/1 for action-based versus topic-based intents.** Though topic-based intents are less successful for classification, they are sometimes necessary for explanation-oriented responses. They are likely to contain training data like 'What does X mean' or 'How do you define X'. In these instances, there would be no strong verbs and only a limited number of ways in which a user would be able to express themselves. Thus, it makes sense to divide your utterances by topic rather than verb. Topic-based intents are also useful for separating topics that you might not want to treat in your VA. For example, utterances relating to religion or death might be topic-based and lead to a straight response from the VA.

- **Keep training utterances short.** Remove lengthy utterances, including any that are longer than two sentences. People generally divide their speech into multiple sentences when they have multiple problems. As such, most intents will be contained within a single sentence. Training data that is several sentences long will likely include multiple main verbs which can confuse the model. For example: 'I am concerned that I might have lost my policy documentation. I would like to make a claim on my phone insurance, can I still make one without the paperwork?' In this

instance, the user is concerned with making a claim and the other information is extraneous. Including the other details in training data would confuse the model and associate the word 'lost' with the word 'claim'.

- **Vary your articles.** Ensure there is a variety of pronouns used across intents. This includes personal, definite and indefinite pronouns. It is also advised to include utterances without pronouns from time to time. For example: 'How do I go about getting *my* new phone', along with 'Need new phone', and 'Want to talk about getting *a* new phone'. In each case the pronoun has changed along with the utterance structure.

- **Vary word order.** Train your model on multiple forms of sentence structure. For example, 'I was arranging new phone insurance when my application failed' could be substituted for 'My application failed when I was arranging new phone insurance'. Your model does not only recognise individual words, but also the structure of individual statements.

- **Vary verb and noun forms.** For every verb that you include within an intent, the noun form and the adjective form should also be added. For example, the intent #RejectApplication would be strongly associated with the verb 'reject', but also ought to contain the noun 'rejection' once, and the adjective 'rejected' once. The verb that is most strongly tied to the intent ought to be used most, with synonyms used less often. In this instance, 'deny' and 'refuse' might also appear in training data. However, it is important to remember to keep a representative balance in your verbs. Specific verbs should not be over-trained as this will lower the confidence of your model for utterances containing other verbs.

- **Vary utterance length.** Some users will type in full sentences, and others in key words. It is necessary to have some variation in utterance length in order to respond to both types of user. The confidence will be lower if a user's text contains only some of the words in training data, even if these words are contained in a full utterance. For example, 'Rejected application' will not match 'Can you tell me more about my rejected application' with a high confidence.

- **Keep one intent per utterance.** Utterances that contain multiple intents should be divided or deleted. An example of such an utterance might be: 'I'd like to submit an insurance claim because someone stole my phone'. This could easily be divided into two intents: #ClaimInsurance and #StolenPhone. However, in cases where there is no main verb in the second intent, it is not necessary to split the utterance apart. For example, 'I'd like to submit an insurance claim for my phone, mine was stolen', could be included in training data for #ClaimInsurance.

- **Remove utterances with multiple main verbs.** This is important even if the second main verb is not another intent as it contributes to noise within the training data. This can be seen in examples such as: 'I would like to make a claim because I recently bought a new phone and it turns out it was a scam'.

- **Remove encoding errors.** Sometimes utterances can be corrupted between systems or fonts and can cause erroneous characters to appear in training data. It is important to remove these before your model is trained.

- **Remove greetings and other 'verbal noise' from the beginning and ends of utterances.** Words such as 'hello', 'please', and 'thanks' should only appear when integrated into an utterance. For example, 'Can you please tell me how to renew my insurance' could be included, though 'Please can I renew my insurance?' would need to be edited before being added into training data.

These recommendations ought to assist you in reducing and refining your training data. Utterances should not simply be added to intents, but should be cleaned beforehand to ensure that the model benefits from their addition.

Creating Entities

Whilst intents are integral to classification, their value can extend beyond this. They don't just have a role in dialog flows, but also impact the weighting of words within training data (see resting entities). Words that have been designated as entity values are given a stronger weighting within the model and consequently boost the confidence of an intent that is tied closely to a given entity.

It is fairly straightforward to create a resting entity in Watson. First, select an intent and establish the major nouns which appear within its training data. In an intent #BrokenPhone, numerous synonyms for the word 'phone' would appear, including words like 'mobile' or 'telephone'. However, as 'phone' would be the most commonly used word, it is this that should be created as the entity value. All variations on the word 'phone' should then be added as synonyms within the entity.

It is useful to bracket entity values in terms of their meanings rather than in terms of direct synonyms. For example, an entity value relating to travel insurance might be defined as 'travel', though its synonyms include words like 'abroad' or 'overseas'.

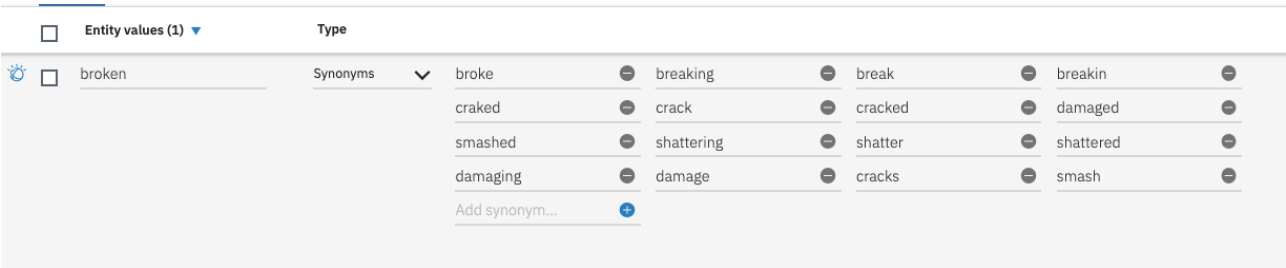
It can often be helpful to check Watson's entity recommendations for a more detailed list of synonyms. Watson Assistant uses the automated synonyms until an entity is built, at which point the automated variations are overwritten. It is generally a good exercise to

check that you have included all possible options at the point at which you create your own entity.

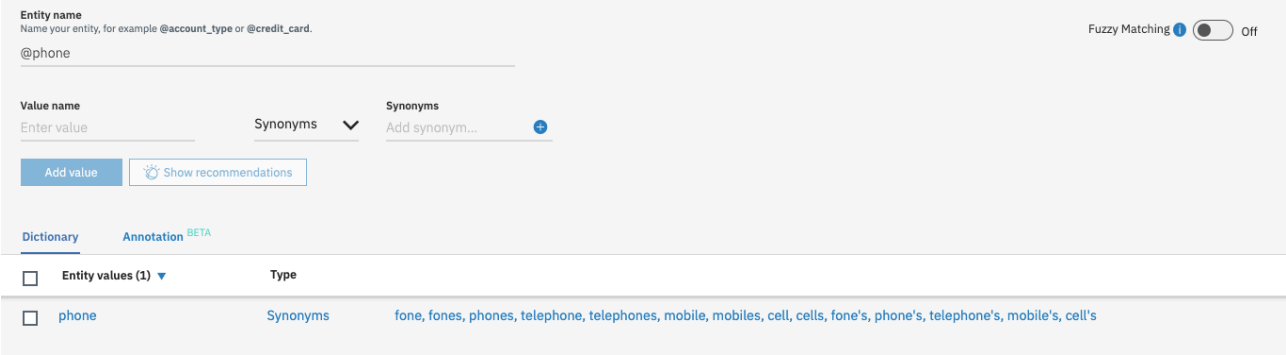
Entity values should generally be the adjective form of a word, such as ‘cancelled’, whilst synonyms ought to follow the noun form, such as ‘cancellation’. If you want to use verbs in your entities, all verb forms should be added too.

Next, it is important to include the plural forms of words as synonyms for entity values. Similarly, common misspellings should be added as synonyms of entity values. The aforementioned entity @cancelled consequently might include ‘cance’ and ‘canceled’. Never add misspellings into training data for verbs, nouns and adjectives as this will confuse your model.

An example entity for #BrokenPhone might look like the example below:



Finally, it is important to alter the major nouns in your training examples so they all correspond to the relevant entity values and not the synonyms. In the #BrokenPhone intent, all utterances would use the word ‘phone’ and not variations like ‘mobile’. As most users would use the word ‘phone’, the overall confidence of the intent will increase. The model will still understand synonyms of entity values: they do not need to be entered as training data. It is better not to dilute training data with artificial utterances that are not representative of the way users speak. Your intent’s training data would consequently look something like the below:



The intent’s training data would consequently include extensive variations on the word ‘broken’ and only one form of the word ‘phone’. It would confidently match most

statements relating to broken phones, and match statements with a lower confidence if the user were not to use the word 'phone'. This is not to say that the confidence would be excessively low, but that it would not have the boost designed to benefit the entity values. It is of course possible to make multiple entity values if there are numerous common ways of expressing a noun.

Iterating & Improving

This section details:

- how to align model performance to business metrics
- a process to perform exploratory analysis on training data, and
- a process to train intent models effectively.

It's written to work very closely alongside the Intent Training Tools repository[8], which includes notebooks to guide you through the Exploratory Analysis and Targeted Training processes using your own WA instance.

A Note on Assistant Plus & Premium

Features such as Intent Conflict Resolution and Intent Recommendation which are available in the Plus and Premium WA plans can perform similar roles to some of the tools described in this section and available in the accompanying notebook. In this section we aim to provide a set of tools which can be used regardless of your WA plan, and will most likely be used before you make any changes directly in Watson. From experience, we've found that it's beneficial to use both this set of tools and the Watson Plus/Premium features in tandem.

The Role of the Model

We can wax lyrical about model performance, but it's often vital to be able to relate an improvement in intent performance metrics back to a client's business metrics. If there's a business metric related to the understanding of the VA this relationship is straightforward. Otherwise, the following approach can be used:

1. **Provide clear and objective definitions to the Business Metrics/KPIs.** All clients define these slightly differently making comparing business performance like-for-like is not as easy as it may seem. E.g "Containment" where the VA has correctly classified the intent, and the Customer requires no further interaction in another

[8] https://github.ibm.com/BusinessDynamics/intent_training_tools

channel to resolve their query. Without validation this may often include abandoned conversations.

2. **Break down the business metric into smaller measurable parts,**E.g. containment may be made up of handoff to an agent by design, escalation due to safe words such as swear words being used, or handoff because a query was not understood by the VA.

3. **Work out which of these parts is affected by model performance.** For the above definition of containment, this would be where escalations occurred due to queries not being understood.

4. **Use this to inform the training process and communicate its success.** You can use information about escalations due to certain intents not being understood to inform the targeted training process. If you can then make a measurable improvement to these intents that is reflected in the business metric, this itself will communicate the value of the training work back to the business.

Exploratory Analysis: Positioning the Starting Block

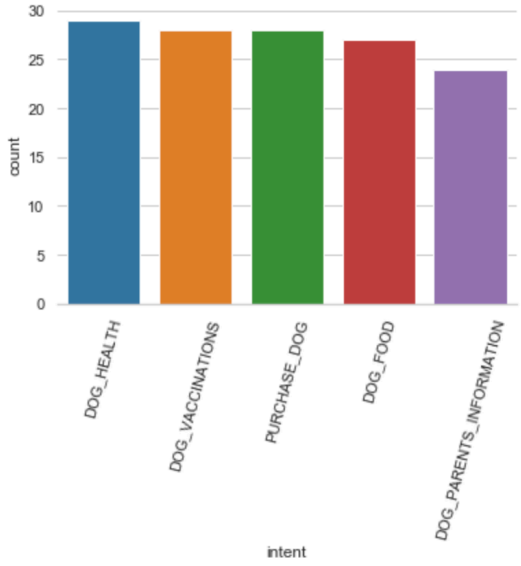
Before attempting to improve any model, it's common practice to perform an exploratory analysis to understand its characteristics and begin thinking about changes that could be made to the data to positively affect the model.

If you are new to the data, haven't delved into it in a while, or it's been built up over an extended period of time, the first thing to do is read through the contents of each intent and entity. While you're doing this, you should be thinking about how well it aligns to the Training Data Selection Best Practices detailed previously in this paper, and looking at user treatment for the intent to ensure that each example in training corresponds to a suitable response. Don't spend ages on this, as you'll later use tools and plots to further explore the corpus.

For the rest of this section we'll describe some example analyses you can perform to understand your training data. It's strongly recommended that you try these out yourself using the Exploratory Analysis notebook[9].

a) Measure Intent Sizes

As a general rule of thumb, your intents should be about the same size as the scope you expect them to cover. If your largest intent contains several times more utterances than your smallest, it will be much 'greedier'[10].



You can check intent sizes using the notebook, or a pivot table and plot in Excel if you don't have access. In the example to the right all intents are of very similar size, so no action is needed, assuming they all have similar scope.

If an intent is much larger than you'd expect based on its scope, you should check for redundancy. This may appear in the form of the same phrase used across multiple utterances, or the same utterance repeated but talking about different nouns.

Action:
Check for redundancy and consider removing it, by modifying or removing utterances, or replacing repeated nouns with entities.

b) Find Representative Ngrams

You should have already read through the examples in each intent - the next stage is to perform statistical analysis to find the ngrams which are most representative of each intent.

In the notebook (below) we use a technique called the *Chi-Squared Test* to measure the dependence between each ngram in training and its class label, returning the 1- and 2-grams which are most strongly associated with that intent.

	DOG_FOOD	DOG_HEALTH	DOG_PARENTS_INFORMATION	DOG_VACCINATIONS	PURCHASE_DOG
top unigrams	dogs, sell, feed, eat, food	chow, night, health, sick, tell	meet, mum, parents, father, mother	injections, vaccinated, shots, jabs, vaccinations	pricing, purchase, want, chinook, buy
top bigrams	puppy eat, dogs food, food sell, dogs eat, dog food	pups happy, dogs depression, dog properly, dog ill, tell puppy	meet father, father pedigree, mother litter, look parents, puppy father	vaccinate dogs, puppies vaccinated, shots dogs, puppy injections, jabs puppy	buy puppy, want buy, buy chinook, like buy, buy dog

[9] https://github.ibm.com/BusinessDynamics/intent_training_tools/blob/master/exploratory_analysis.ipynb
[10] We've found 'greedy' to be a business-friendly way of referring to low precision – see Developing Informative Metrics

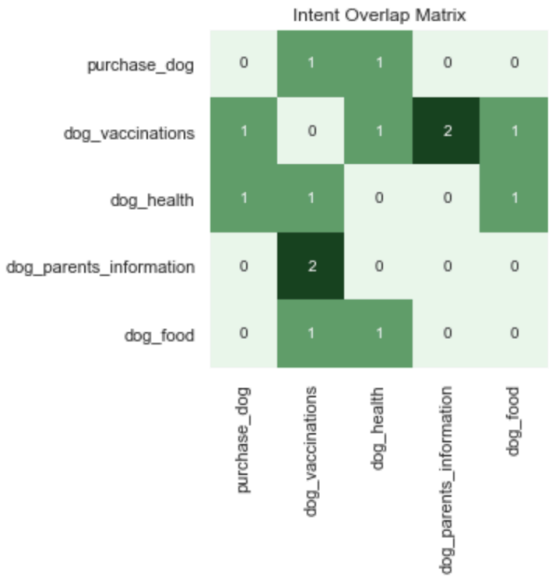
Looking through the list of intents, you can see that we expect most of these correlations to appear. ‘Sell’, ‘feed’, ‘eat’ and ‘food’ are all strongly associated with the intent DOG_FOOD: this is a good thing!

This analysis also shows, however, that the word ‘dogs’ is strongly related to the intent DOG_FOOD, and ‘chinook’ is strongly related to PURCHASE_DOG. It would be wise to pull out a *dog* entity containing values including breeds such as ‘chinook’, to prevent this imbalance affecting intent classification in the dialog flow.

Action:
Check that you’d expect each unigram and bigram in the table to be tied to its corresponding intent. If not, consider rebalancing the intent, removing problematic examples, or adding generic ngrams (such as ‘dog’ above) and their synonyms (‘chinook’, ‘terrier’) as entity values. If you add a generic ngram as an entity, ensure that it’s evenly distributed across all intents in the workspace so the added weight doesn’t bias the word ‘dog’ towards one intent.

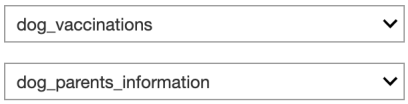
c) Inspect Intent Overlaps

A final piece of analysis that can be done in the exploratory phase is to inspect the intents in a skill that share phrases with each other. A tool in the exploratory analysis notebook creates a matrix, similar looking to a confusion matrix, which shows the number of shared ngrams between different intents.



By default, the matrix (right) shows the number of times a 4- or 5-gram appears in utterances in two different intents. Darker coloured sections of the matrix indicate pairs of intents which have multiple similar utterances, which are likely to cause confusion in intent classification.

The notebook then enables you to then look at the ngrams which are shared between two intents, and search for the utterances containing these ngrams.

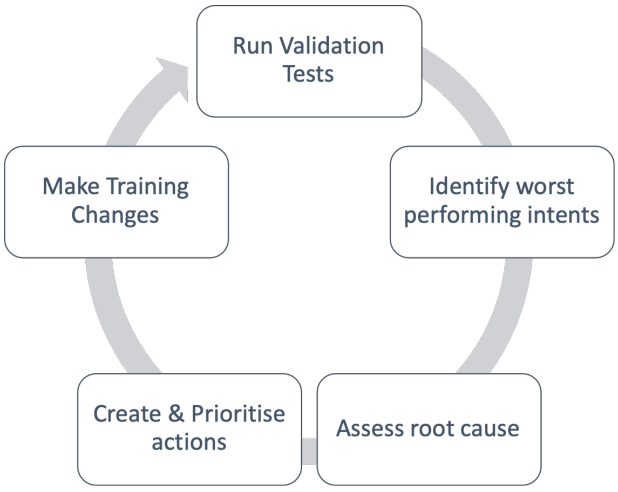


	dog_vaccinations	dog_parents_information
can i check if	1.0	1.0
i check if my	1.0	1.0

Action:
Inspect the intent pairs with greater semantic overlaps, drilling down to the ngrams and the utterances that contain them. We'll work on prioritising and resolving these changes in the next section.

Improving the Model: Targeted Training Approach

This section describes a tried and tested approach to training intent models, which brings the previously described best practices, tools and testing methods together in a method that can be used for continuous improvement. The key idea behind this approach is ***not to try to tackle the whole skill at once***. Instead, we run rigorous tests to identify specific intents which are both low performing, and for which we can identify actions to reliably improve their performance.



Targeted training method.

We'll describe the process below using the Targeted Training notebook[11] in the Intent Training Tools repository. It's strongly recommended that you try this out on a skill of your own.

1. Run Validation Tests

In order to identify the intents to target, the first thing we need to do is run validation tests. If you're lucky, you might have a validation set you can use for this step; however, we'll stick to K-fold here[12].

You can run K-fold tests using the tool in the Intent Training Tools repository, or directly from the notebook. Both methods will return two outputs: the result for each utterance, and metrics for each intent. If you run tests using the notebook you should see an output similar to below.

```
In [4]: import logging
logger = logging.getLogger()
logger.setLevel(logging.CRITICAL)

kfold = kfoldtest(apikey=apikey, url=url, n_folds=nfolds, threshold=threshold)
kfold.intent_df_from_df(train_df)
results_kfold, classification_report = kfold.full_run_kfold_from_df(train_df)
```

```
[2019-07-30 13:14:52,023] INFO (kfoldtest): You have space to perform the k-fold test
[2019-07-30 13:14:52,028] DEBUG (kfoldtest): fold num 1: train set: 89, test set: 47
[2019-07-30 13:14:52,029] DEBUG (kfoldtest): fold num 2: train set: 91, test set: 45
[2019-07-30 13:14:52,030] DEBUG (kfoldtest): fold num 3: train set: 92, test set: 44
[2019-07-30 13:14:52,033] INFO (kfoldtest): Creating kfold workspaces..
[2019-07-30 13:14:55,961] INFO (kfoldtest): Checking workspaces..
[2019-07-30 13:15:19,124] INFO (kfoldtest): Checking workspaces..
[2019-07-30 13:15:20,195] DEBUG (kfoldtest): Workspace 1 available
[2019-07-30 13:15:22,142] DEBUG (kfoldtest): Workspace 3 available
[2019-07-30 13:15:42,148] INFO (kfoldtest): Checking workspaces..
[2019-07-30 13:15:43,098] DEBUG (kfoldtest): Workspace 1 available
[2019-07-30 13:15:44,142] DEBUG (kfoldtest): Workspace 2 available
[2019-07-30 13:15:45,281] DEBUG (kfoldtest): Workspace 3 available
[2019-07-30 13:16:05,288] INFO (kfoldtest): Running test for fold 1
100%|██████████| 47/47 [00:47<00:00,  1.03it/s]
[2019-07-30 13:16:52,674] INFO (kfoldtest): Running test for fold 2
100%|██████████| 45/45 [00:43<00:00,  1.11it/s]
[2019-07-30 13:17:36,672] INFO (kfoldtest): Running test for fold 3
100%|██████████| 44/44 [00:47<00:00,  1.08s/it]
[2019-07-30 13:18:24,361] INFO (kfoldtest): Deleting temporary workspaces
```

2. Identify Worst Performing Intents

The intents which you want to target are likely to be the lowest performing ones. Follow the steps in Developing Informative Metrics to choose whether precision, recall, accuracy or F1 score is the best measure of intent performance for your skill, and remember that this may not be the same for every skill in your VA. You may also want to prioritise intents based on a certain business need, as described in Deciding the Role the Model Plays.

[11] https://github.ibm.com/BusinessDynamics/intent_training_tools/blob/master/targeted_training.ipynb

[12] In practice the results for K-fold tests can vary, as they only use one random sample, split across K folds. When implementing this method we've found it more effective to take the average of at least 2 K-fold tests for each step.

The notebook helps you to prioritise intents based on your chosen metric. The intents ranked by performance in your chosen metric are exported to a CSV so that you can make comments and annotate actions. The below images show what you'll see in the notebook.

Now, we need to choose which metric to prioritise for this workspace.

If FPs and FNs are equally negative, then you may want to look at F1 score. More on this in **Section x** of the training at scale guide.

Preferred metric:

precision

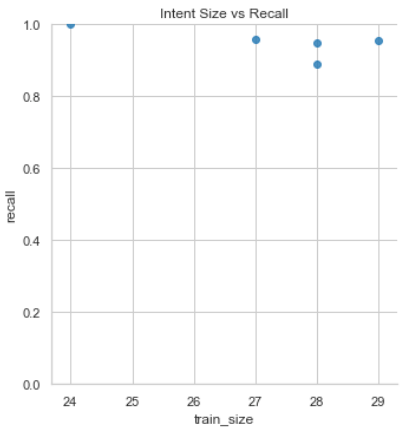
We can then view K-fold results sorted by this metric, with a column added to show the size of the training set, and a couple more for annotation.

These results have been exported to /Users/kalyan/Documents/practice-cog/intent_training_tools/results/puppy_kfold_results_20190731-1010.csv for annotation throughout the targeted training process.

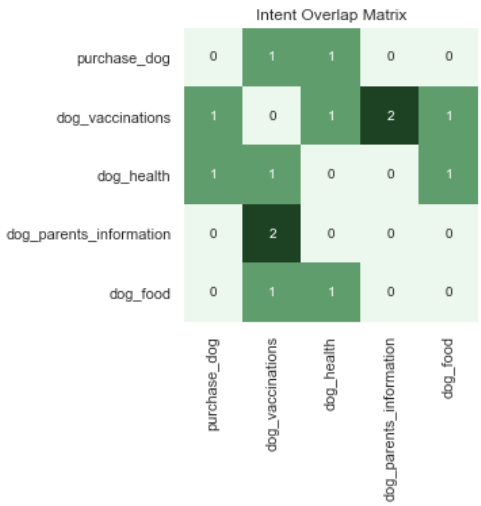
confusion	threshold	accuracy	precision	recall	F1	train_size	comment	action
DOG_HEALTH	0.4	0.718750	0.733333	0.956522	0.830189	29		
PURCHASE_DOG	0.4	0.757576	0.827586	0.888889	0.857143	28		
DOG_PARENTS_INFORMATION	0.4	0.869565	0.869565	1.000000	0.930233	24		
DOG_VACCINATIONS	0.4	0.857143	0.900000	0.947368	0.923077	28		
DOG_FOOD	0.4	0.925926	0.960000	0.960000	0.960000	27		

3. Assess Root Cause

Intent Size vs Recall. One of the analyses we can perform on our intents is to compare size of training data to recall. A common cause of low recall is that an intent does not have enough similar training samples to an utterance to produce a high confidence. This results in the intent matching but below the confidence threshold, so the user does not receive a response (FN).



We can see from the dog skill that all intents have similar training sizes and high recall, so there are no actions to take here.



Intent Overlaps. Another cause of poor intent performance can be confusion between intents. Potential for confusion can be identified from phrases that are commonly used in more than one intent, as shown in the matrix to the left.

It's important to note the assumed weighting of different words when looking at overlaps - more on this in Intents & Entities: Under the Hood.

The notebook allows you to look at the shared phrases for each intent pair and to search for training samples containing these phrases.

Directly from Classification. If you've run a validation set, you can also try to find training samples that may have caused misclassification of a particular utterance. You can do this by searching the training corpus for words or phrases which are present in training data. You may also want to try different forms or orders of the verbs or nouns present in the utterance if your initial search doesn't return any useful-seeming samples.

3.3 Diagnose Confusion

You can also find training utterances that may have caused misclassifications, by entering an utterance that has been misclassified. Note that the list produced is only a guide which will find the most obvious samples (no stemming or lemmatisation), and in no way exhaustive.

You should also use the table above to search for different forms of words in the utterance.

i want to get a cat

	utterance	intent	ngrams found
134	What sort of food do I need to get to keep the puppy happy?	dog_food	'to get'
58	I want to get my dog a check up urgently. He don't seem the same	dog_health	'i want', 'want to', 'to get', 'i want to', 'want to get', 'i want to get'
63	Was the litter sick for the puppy I want to get?	dog_health	'i want', 'want to', 'to get', 'i want to', 'want to get', 'i want to get'
97	I want to know more about their parenting	dog_parents_information	'i want', 'want to', 'i want to'
41	my poodle had it's first set of vaccs last week and I want to get the next set, when should I do this	dog_vaccinations	'i want', 'want to', 'to get', 'i want to', 'want to get', 'i want to get'
43	I want my puppy to be protected against disease	dog_vaccinations	'i want'
31	at what age do I not need to get my pooch jabs	dog_vaccinations	'to get'
33	I need to get some injections for my drakehead lab	dog_vaccinations	'to get'
0	I want to buy a minature schnauzer	purchase_dog	'i want', 'want to', 'i want to'
5	I want a dog	purchase_dog	'i want'
8	i want to get a puppy	purchase_dog	'i want', 'want to', 'to get', 'get a', 'i want to', 'want to get', 'to get a', 'i want to get', 'want to get a'
10	No I wanted a puppy.	purchase_dog	'i want'
11	I want to buy a chinook	purchase_dog	'i want', 'want to', 'i want to'
26	i want to add a dog to the family. How much would it cost	purchase_dog	'i want', 'want to', 'i want to'
14	Can I get a puppy?	purchase_dog	'get a'

A tool to automate this process is available in the targeted training notebook.

4. Create & Prioritise Actions

You should have enough information from the previous step to decide on some actions to improve model performance. These should fall into one of the following categories:

- Resolve a clash, or multiple clashes.
- Clean training data.
- Get more training data.

It's advised to use the action column in CSV which was exported from the K-fold test to note down the actions you're planning on taking.

The best way to prioritise actions is to predict the size of their impact. Is the clash large? Do the utterances within the poor performing intent align to the best practices? Is the volume of chats in the live VA for that intent great enough for you to get data?

5. Make Training Changes

This is where most of the work is. After doing this a few times you'll develop your own intuition for what works best for your VA, but at first you may want to run K-fold tests more regularly to see the effect of your changes.

Don't be shy about being bold with your changes at this stage – a user never sees the training data, only its effect on model performance. Although, when you're making changes, ensure that the treatment for each intent is what you'd expect for every training utterance in that intent.

Below are some tips for making changes to training data.

Resolving clashes	Prioritise phrases that are likely to be weighted highly by the intent classifier. Identify whether shared phrases should be specific to one intent. If there's a common phrase that's shared, try to achieve a balance across all applicable intents.
Cleaning training data	Follow the advice in Deciding Utterances. Think about intent size relative to other intents.
Getting more training data	You'll need access to real user messages for this. Try to get them in a format where they're searchable, and you have their predicted intents and confidences. If you have a pipeline where you can remove utterances produced from in-VA button clicks as well as utterances already in training data, even better. There are two methods for this: • 'guided' learning: search for utterances containing phrases that you'd expect to appear in an intent. • 'active' learning: prioritise addition of utterances where the difference between the confidence for the first intent and confidence for the second intent is small. This is the classifier's best guess at utterances that it might've misclassified. It's often best to combine the above approaches. Modify utterances when adding them as training, using the advice in Deciding Utterances.

6. Test (Again!)

To show the effect of your training work, run a validation test after each of the actions you complete. These can be used to communicate success of training to stakeholders, and the changes should be tested again as part of the release process before entering the production VA.

Conclusion

This document should have given you most of what you need to be able to manage the models powering a Virtual Assistant at any scale. We've covered best practices from designing from scratch, from an architectural level to an utterance level. We've also covered approaches you can use for an existing implementation, or to keep your implementation at its best as it keeps growing.

As with any developing technology the tools and methods for managing VAs, not to mention the features of the Watson Assistant product itself, are constantly changing. We've tried to make the ideas expressed here as agnostic to these changes as possible, and will endeavour to keep publishing our learnings as a practice as the VA landscape develops.