Enhancing Intent Analysis with Context

Joseph M. Wills

Enhancing Intent Analysis with Context

Joseph Wills
Intended graduation: May 2022
Computer Science (B.S.)

Research Advisor: Dr. Joe Near
Last date of revision: 7 May 2022

1. Introduction

1.1 Overview

Intent analysis is a major node in the intersection between advances in artificial intelligence and its applications in the lives of users. Conversational agents that analyze user intent from spoken or typed utterances are used in general use cases, such as the iPhone's Siri or the Android Google Assistant, and specific topic areas, such as a digital assistant on a bank's website. Intent analysis is the problem in computer science of taking a user utterance and predicting the user's intent.

One specific hurdle facing modern intent analysis solutions is that they are, for the most part, unaware of specific user context. Some personal assistants -- like Google Assistant -- can offer limited information based on context, such as giving updated driving routes based on traffic, but none currently use the specific user's context to enhance their intent analysis predictions.

For example, a user may input this utterance: *I would like to change office, please*. An existing intent analysis solution may predict, with equal weight, two user intentions:

1. "I would like to reschedule the office meeting today."
2. "I would like to update Microsoft Office."

Without user context, it is impossible for the computer to accurately predict the correct user intent. This research project aims to fill this gap by developing a context analyzer for intent analysis. In this example, if the computer knew the user was currently looking at their calendar, it would weigh the "I would like to reschedule the office meeting today" intent higher than the other one; conversely, if the user was looking at Microsoft Word, it would give higher weight to that intent. The goal is to enhance intent recognition by including deeper information about what the user is doing.

The aim of this research, put simply, is to use contextual information to improve intent analysis.

2. Background

There are many examples in the literature of new advancements to the state of the art of intent analysis, such as zero-shot learning schemes (Luong), or ones that merge sentiment and intent analysis (Lin), for example. Iterating upon existing intent analysis solutions is not a goal of this research project; instead, it is to create a classifier to enhance the output of existing intent analysis solutions.

User modeling. Part of this project is concerned with modeling specific users based on their individual use contexts. This has been explored to some extent in the literature (Guha), particularly by researchers at Google. However, their user modeling approach is not applicable to this project: their intent is specifically to offer the “right information at the right time, without the user querying for information,” and this project is focused on enhancing intent analysis, which requires user queries. However, certain suggestions -- such as tracking web search traffic -- are useful, and Google is the most up-to-date in terms of consumer software that recognizes user context. Google as a whole does use context like previous web searches to provide information tailored to the user.

Statistical analysis of accuracy. Most publically available datasets for testing user utterances focus on querying information repositories such as Wikipedia for “factoids” or otherwise lists of details about certain target subjects.[1] The lack of a publicly available corpus of ambiguous utterances led to the creation of our own set of test utterances.

3. Terminology

In order to understand this project, a number of useful definitions can be employed:

1. **Intent analysis.** Intent analysis is the process of analyzing a piece of text – the user "utterance" – for the writer's intention, and is used in a variety of applications. In this case, we are interested in intent analysis as it relates to conversational agents (for example, Siri or Google Assistant). Many papers have been released on the subject in both general and specific domain areas. Improving the actual intent analysis produced from text using natural language processing is not the goal of this project; rather, it is to use external information (context) to enhance an existing intent analysis solution. As such, this project will use an existing, publicly available intent analysis solution.
2. **Slot-filling.** In intent analysis, slot filling is the process of taking objects referenced in an utterance and filling them into "slots" based on intent. In the utterance, "Send a text to Joe," slot-filling would be setting the recipient slot to "Joe". Slot-filling is not a target area of improvement for this project.
3. **User context.** Various other methodologies have been used within the literature to analyze user behavior. For our purposes, context or user context is referring to the specific "snapshot" of user context at any given moment, which consists of two pieces of data:
    a. The amount of time spent on each program in the last five minutes
    b. The currently active program

[1] https://trec.nist.gov/pubs/trec13/papers/QA.OVERVIEW.pdf

4. Methods

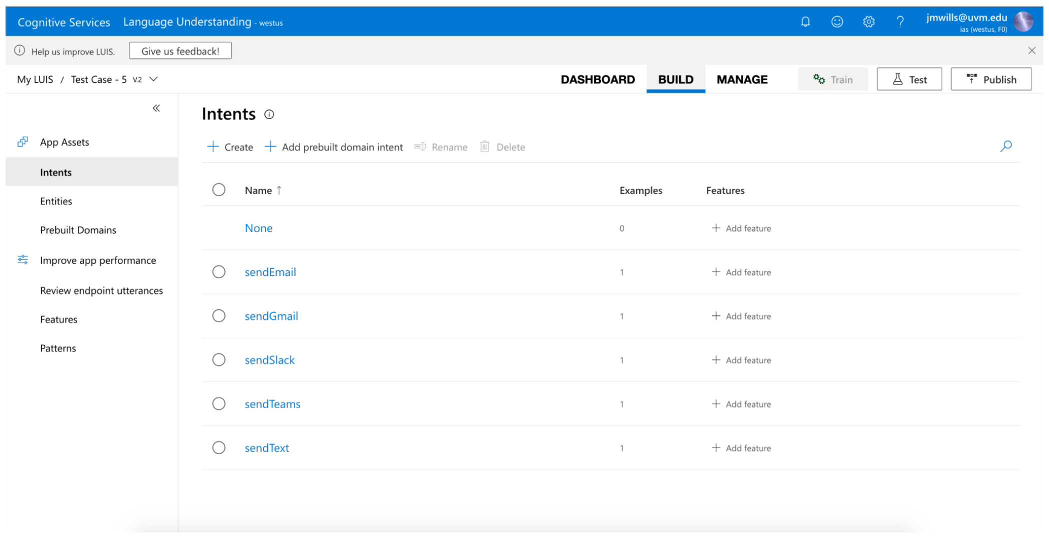
4.1 Project description

The major component to be built for this project is a context analyzer, one which ingests usage context information and weighs a set of intents accordingly. This is the major original design component of the project; a third-party intent analysis platform, Microsoft's LUIS.ai, will be used to generate non-contextually enriched intent analysis predictions to be used during testing.

An important note is that it is not part of the end goal of the project to create a realtime conversational agent; the goal is to determine that context aids in intent analysis, not to necessarily create a personal assistant that leverages that fact (although these findings could easily be applied to one).

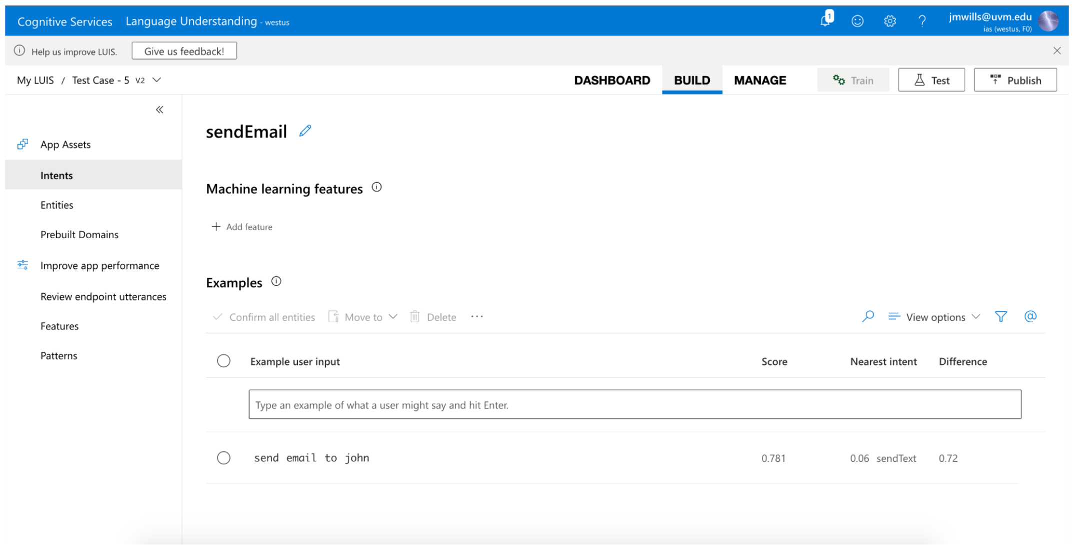
4.2 Notes on intent classification

LUIS.ai works (like most enterprise intent analysis solutions) by building apps that each have their own trained intents, which in turn are based on their own set of utterances. To be clear, this is an example of the form a LUIS.ai intent analysis app may take:


Cognitive Services Language Understanding - westus
jmwills@uvm.edu
Help us improve LUIS. Give us feedback!
My LUIS / Test Case - 5 V2
DASHBOARD BUILD MANAGE
Train Test Publish
App Assets
Intents
Entities
Prebuilt Domains
Improve app performance
Review endpoint utterances
Features
Patterns
Intents
Create
Add prebuilt domain intent
Rename
Delete

Name ↑	Examples	Features
None	0	+ Add feature
sendEmail	1	+ Add feature
sendGmail	1	+ Add feature
sendSlack	1	+ Add feature
sendTeams	1	+ Add feature
sendText	1	+ Add feature

Where each intent has its own dashboard, containing examples of utterances that LUIS.ai will use when evaluating intents.



Example user input	Score	Nearest intent	Difference
Type an example of what a user might say and hit Enter.			
`send email to john`	0.781	0.06 sendText	0.72

Intents can be cached – it is not necessary to *call* LUIS.ai's API every time you evaluate intent, because the same input utterances will receive the same API response.

4.3 Data and data collection

Data was collected using an observer that collects data on user behavior as described above:

c. The amount of time spent on each program

d. The currently active program

After five minutes, these data points would be stored and reset, so effectively we are considering five minute chunks of user activity at a time. For example:

Mail	Messages	MSTeams	MailisOn?	MessagesisOn?	MSTeamsisOn?
40.3	50.0	12.1	1	0	0

("isOn" columns correspond to which program was active at that time.)

In a realtime solution, the observer would look back at the last five minutes dynamically – this observer is used to gather training data.

4.4 Problem domain

The problem domain was determined to be Communication to narrow the scope of the project. Target intents – that is, intents that we intend to differentiate between – were chosen based on what would likely be ambiguous for the intent analyzer to understand. For example, the first test case, described in more detail below, uses intents like these, with corresponding example utterances:

Intent	Example utterance
sendEmail	send email to john
sendText	send text to john

4.5 Labeling

Context information was labeled by row based on subjective analysis of what at the time would make sense for the user to want to do. So for this example

Mail	Messages	MSTeams	MailisOn?	MessagesisOn?	MSTeamsisOn?	Intent
40.3	50.0	12.1	1	0	0	sendMail

sendMail was determined to be the proper intent because the user a) spent a lot of time in Mail, relatively, and b) was actively using Mail at that time. For the domain of Communication, this makes sense, because someone would likely be meaning to communicate using the program they have up on their screen. Further discussion of this follows in other sections.

4.6 Evaluation

Evaluation of intent classification problems is tricky; to truly evaluate whether or not something works, it essentially comes down to asking "did it properly evaluate the user's intent?" which is thereby ultimately asking for an impossible metric, i.e to read the user's mind to tell if their intent was executed properly. It is important to note that because of this, the intent was subjectively determined, so the tests were done on that basis.

4.7 Classification approach

A number of different classifier strategies were used, which are discussed more later, but the major different pathways were the use of a kNN classifier versus a Naive Bayes classifier, and different pre-processing of the data was used, which is discussed more in the next section. Ultimately the classifier was meant to demonstrate that context classification is possible; approaches discussed and used here may not make sense when dealing with the problem at production scale.

There were other classifier options that were not tested. The kNN was originally considered because for the tests done, it was a) easy to test with considering time constraints and b) computationally accessible for the amount of data. In fact, because there was a relatively small dataset, a kNN algorithm seemed like a natural choice.

A Naive Bayes classifier is also included, for a few reasons. A Naive Bayes classifier is closer to what a context classifier may use in production because it is highly scalable. In a production implementation of context classification, such as one using supervised learning discussed in section 7.2, a much larger amount of context data would be used compared to the tests in this thesis, so scalability is an important consideration. Conversely, a kNN would perform worse more quickly on large sets of data. Of course, it may likely turn out that neither classifier makes sense for a production context classifier.

A rules based approach was considered and ultimately not used, because a rules based approach seemed ad-hoc, and highly specific to whatever test case was used; while the goal of this project is not to create a totally general purpose context classifier, a rules based approach seemed trivial. Ultimately, the method of data collection may have led to a de-facto rules based approach, which is discussed later.

There are many other choices of classifiers that were not used. We forwent a neural network because the amount of data used was relatively small, so the tradeoffs in time spent designing the neural network and computational concerns were not worth the gain, because other methods were considered to be suitable for these test needs.

Classifiers were also ruled out because this project necessitates multiclass classification, so we focused on classifiers that had accessible multiclass implementations. In the future, it is possible that a context classifier may even necessitate multilabel classification, but that was not a consideration for this project.

5. Results

5.1 Example 1

The first and longest set of test cases were concerned with differentiating between an expanding set of possible utterances.

In this test, the given utterance is “send to john.” LUIS.ai attempts to determine an intent between a number in a range between two and ten:

Intent	Example utterance
sendDiscord	Send discord to john
sendEmail	Send email to john
sendFacebook	Send facebook to john
sendGmail	Send gmail to john
sendSlack	Send slack to john
sendTeams	Send teams to john
sendTelegram	Send telegram to john
sendText	Send text to john
sendWhatsapp	Send whatsapp to john
sendYahoo	Send yahoo to john

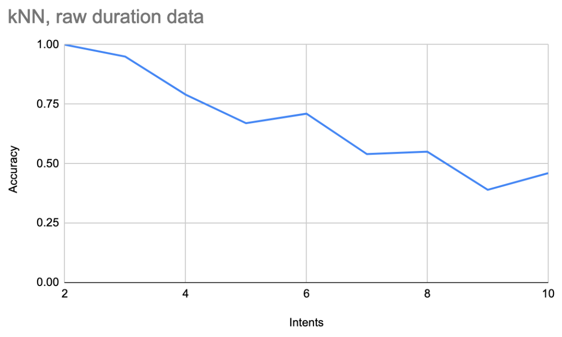
Nine sets of trials were run, beginning with two distinct intents and reaching the way to ten intents. So for the first trial, LUIS.ai only attempted to differentiate between sendMail and sendTeams. The issue is that without any features of the utterance “send to john,” LUIS.ai has no real basis to make a guess as to the user intent; therefore, it is necessary in this case to look at context.

For each of these sets of trials, a classifier was trained to differentiate between the growing number of intents, which were each associated with their own program as exemplified above.

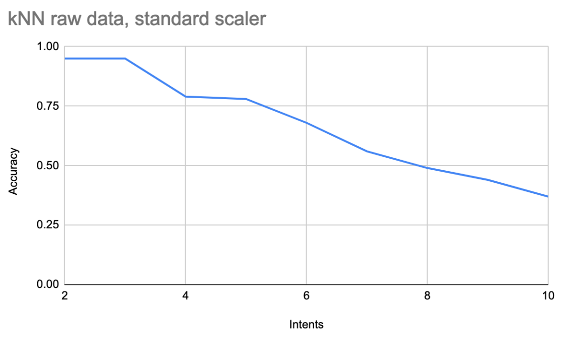
The first classifier trained was on raw time data, like so:

Discord	**Mail**	**Facebook**	**Gmail**	**Slack**	**Teams**	**Telegram**	**Messages**	**WhatsApp**	**Yahoo**	**Intent**
11.18	12.74	4.68	50.17	50.56	23.92	56.41	32.89	34.45	0.13	sendTelegram

For each set of trials, only the data that was relevant to the included intents was used; so a kNN was trained on just the set including rows in [Mail, Messages], then a kNN was trained on the set including [Mail, Messages, Teams], and so on. This process was continued until all ten example intents were included.



The model’s ability to discriminate between intents, predictably, decreases as the number of intents increases. Using a standard scaler in SKLearn – which centers and scales the features – smoothed the curve, but still lost a great deal of accuracy when intents became very high:



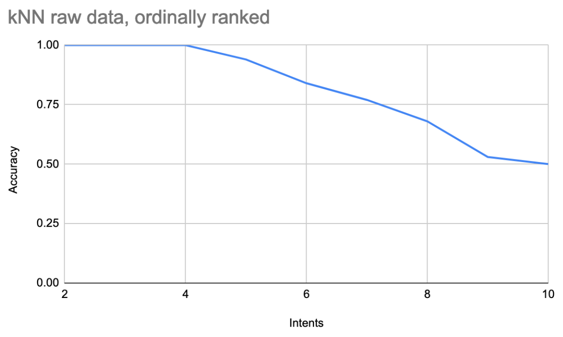
A different approach was then adopted to train the kNN on examples that were not raw time data, but ordinally ranked time data. The program with the highest amount of time spent became 1, the second highest became 2, and so on. So this example:

Discord	Mail	Facebook	Gmail	Slack	Teams	Telegram	Messages	WhatsApp	Yahoo	Intent
11.18	12.74	4.68	50.17	50.56	23.92	56.41	32.89	34.45	0.13	sendTelegram

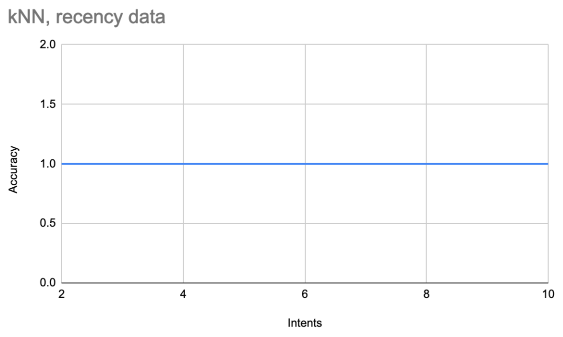
Became this:

Discord	Mail	Facebook	Gmail	Slack	Teams	Telegram	Messages	WhatsApp	Yahoo	Intent
8	7	9	3	2	6	1	5	4	10	sendTelegram

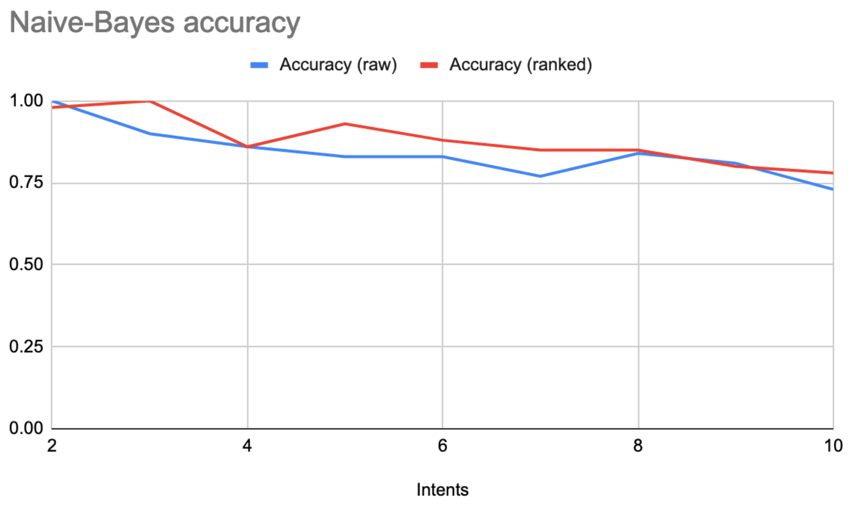
Training on this data yielded better accuracy, especially when intents were n <= 6:



The effect of ordinally ranking this data was to normalize it beyond what the standard scaler effectively did. Using recency data – that is, simply the most recently active program – yielded 100 percent accuracy:



Training a Naive-Bayes classifier on the raw and ordinal time data yielded much stronger accuracy results especially when the number of intents became large, with a much shallower drop off:



Whereas the kNN's accuracy fell to ~.5 by the time the number of intents grew to ten, this Naive-Bayes classifier maintained a relatively strong accuracy throughout.

5.2 Example 2

Another, shorter example was produced using a different set of intents:

Intent	Example utterance
forwardEmail	Forward this message to john in email
forwardGmail	Forward this message to john in gmail
forwardYahoo	Forward this message to john in yahoo
forwardMessages	Forward this text to john in messages
forwardFacebook	Forward this text to john in facebook

The example utterance "forward this message to john" yields the following intent prediction from LUIS:

forwardEmail	forwardGmail	forwardYahoo	forwardMessages	forwardFacebook
0.250468	0.116693	0.13184	0.036694	0.045367

6. Discussion

The initial hypothesis of this thesis – that context could be used to enhance intent analysis – has been confirmed. Especially while differentiating between a low number of similar intents, a context classifier was able to determine intent from context. However, there are several noteworthy observations to be taken from this work.

The data-labeling approach meant that, despite using statistical classifiers for the context classifier, this was essentially a rules-based approach. The post-hoc labeling process led to a system where accuracy was determined based on whether or not the classifier could, essentially, determine which program was used the longest in the five minute interval. Had this rule just been programmed into the context classifier, bypassing a statistical classifier, one can imagine it would have 100 percent accuracy.

However, the first test example is not the whole picture. Each of the example utterances in the first example has one discriminating feature, the name of the program associated with that intent:

Send *yahoo* to john

Send *whatsapp* to john

The second example includes two discriminating features:

Forward this *text* to john in *messages*

Forward this *message* to john in *email*

We can see how this affects intent predictions from LUIS:

utterance: "forward this message to john"				
forwardEmail	forwardGmail	forwardYahoo	forwardMessages	forwardFacebook
0.250468	0.116693	0.13184	0.036694	0.045367

In the first example, the LUIS predictions were essentially worthless, because it lacked any way to differentiate between any intents. In this second example, there is a discriminating feature in the utterance: "message" as opposed to "text." LUIS' weights for "forwardEmail", "forwardGmail", and "forwardYahoo" are arbitrary, for the same reason they were arbitrary in the first example. We can see from this example that a higher dimensionality of differentiating features in the utterance leads to LUIS being able to discriminate within a number of intents, but not all of them, leading to an opportunity for a context classifier to differentiate between the remaining ones.

This suggests that a larger and more complicated intent space may be very complex, and a rules based approach may not work. Ultimately, a totally different approach to gathering data and training may be necessary to make a fully realized context classifier. Another approach could be to gather data on the user as they use a conversational agent over time. As they use their conversational agent, their utterances, current context, and intent analyzer's intent prediction are recorded. Over time, this data can be used to train a context classifier. A mix of supervised and unsupervised learning might be necessary; a large amount of data would make it infeasible to label everything by hand, but there is a problem of knowing whether or not the original intent prediction is correct.

7. Potential Future Work

The scope of this research project aimed to answer the question of if usage context could be used to enhance intent analysis. Looking forward, we can anticipate several problems if building a context-analyzer for scale, and pathways to potentially stronger results.

7.1 Slot filling

One problem that is outside the scope of this project is slot filling, as in determining specific entities present in an utterance. Consider:

1. Send a message to *Alice*.
2. Send a message to *Bob*.

An intent classifier like LUIS can identify entities in user utterances. We did not concern ourselves with classifying entities because it increases the complexity of the problem massively. The issue faced by expanding the current project to include context-based slot-filling is that it requires a much deeper understanding of the behavior of a user in a computer system. The current observer, which tracks programs in use, is comparatively uninvasive when one considers that to effectively fill slots based on context, the context analyzer would have to integrate with all of a user's messaging apps and create cross-platform profiles of their contacts.

Another approach might be to carry over the slot filling predictions from the intent analyzer, which does slot filling on its own based on user utterances. This would allow us to bypass a deeper, more invasive integration with the user system.

7.2 Unsupervised/supervised learning

One confounding problem for context analysis at scale is classification. Across a group of users, the same contexts may be associated with different intents – it is unfeasible to hand label every user context for every user.

A potential fix for this problem is unsupervised learning. A context classifier built for a home computer, for example, could forgo context classification for the first several months of usage or several hundred utterances, and use context classification once it has gained enough data to make context-enhanced intent predictions based on unsupervised clustering of contexts.

Another approach is to use a mix of supervised and unsupervised learning. To create good training data in the approach described above, one would have to also know which predicted intents were ultimately successful.

7.3 Clarifying questions

Conversational agents frequently employ clarifying questions to differentiate between two or more ambiguous intents; for example, if a user asked Siri to turn off an internet connected lightbulb by saying "please turn off the light," Siri may respond by asking if they mean the kitchen or living room bulb. Clarifying questions could also be employed in the larger scale learning approaches described in sections 6 and 7.2, to get data on whether or not the non-context enhanced utterance was correct.

7.4 Domain specific weights

In this project, we focused on one single intent domain – communication – for the intents. In some tests, the scores inputted to the classifier were weighted by a set of complimentary data that included whether or not a program was open at that time. For communication this makes a lot of sense: if we are differentiating between, say, Microsoft Teams and Slack, someone is more likely to be trying to interact with the one that is currently open. Conversely, this may not be true in another domain; it may turn out that this association is less strong when, say, a user is using a PDF reader, and we may find in further tests that weights that take the form of binary, like this:

Program 1 is On?	*Program 2 is On?*	*Program 3 is On?*	*Program 4 is On?*
1	0	0	1

may be more effective in some domains if we implement decimal point values for weights, like this:

Program 1 is On?	*Program 2 is On?*	*Program 3 is On?*	*Program 4 is On?*
.4	0.1	.2	.8

Conjecturally, domains that would fit an approach like this would be ones where multiple contexts are in use at the same time; for example, this approach could be used to differentiate media control utterances like “start music” or “stop music” based on what apps were recently open.

8. Conclusion

This project set out to determine whether or not context could be used to enhance intent analysis predictions; the answer is yes. While much more work is needed to create a general purpose, flexible context classifier, this project also produced important takeaways about the nature of this problem and pathways for further exploration.

Citations

Guha, R., Gupta, V., Raghunathan, V., & Srikant, R. (2015). User Modeling for a Personal Assistant. *WSDM '15: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. doi:WSDM '15: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining

Lin, Chenghua, and Yulan He. 2009. “Joint Sentiment/Topic Model for Sentiment Analysis.” Pp. 375–84 in *Proceedings of the 18th ACM conference on Information and knowledge management, CIKM ’09*. New York, NY, USA: Association for Computing Machinery.

Manning, A., & Danqi, Y. (2017). Position-aware Attention and Supervised Data Improve Slot Filling. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 35-45.

Thai-Le Luong, Minh-Son Cao, Duc-Thang Le, Xuan-Hieu Phan, "Intent extraction from social media texts using sequential segmentation and deep learning models", *Knowledge and Systems Engineering (KSE) 2017 9th International Conference on*, pp. 215-220, 2017.