Proceedings of the Twenty-Ninth International
Florida Artificial Intelligence Research Society Conference

Prioritization of Risky Chats for Intent Classifier Improvement

Ian Beaver and **Cynthia Freeman**
NextIT Corporation,
12809 E Mirabeau Pkwy, Spokane Valley WA 99216 USA
http://www.nextit.com

Abstract

When reviewing a chatbot's performance, it is desirable to prioritize conversations involving misunderstood human inputs. A system for measuring the posthoc *risk of missed intent* associated with a single human input is presented. Using defined indicators of risk, the system's performance in identifying misunderstood human inputs is given. These indicators are given weights and optimized on real world data. By application of our system, language model development is improved.

Introduction

NextIT is a company in Spokane, WA, USA that deploys chatbots in several domains where they interact with human users. These chatbots will be referred to as *virtual agents* as they perform tasks essential in the customer service realm. These virtual agents map user inputs, or *chats*, to a *derived intent*. In the context of Natural Language Processing, intent is defined by Dumoulin as *"an interpretation of a statement or question that allows one to formulate the 'best' response to the statement"* (Dumoulin 2014). There are multiple approaches to intent recognition, such as [(Cohen, Morgan, and Pollack 1990), (Holtgraves 2008), (Montero and Araki 2005)], but we assume the pre-existence of chatbots which perform such tasks.

The collection of rules that defines how input language maps to an intent is referred to as a *language model* within this paper. While there exists methods to introduce confidence scoring within language models used by speech recognition systems, they rely on features present in the acoustic models, word lattice density, etc [(Pradhan and Ward 2002), (Wessel et al. 2001)]. Our topic is that of measuring posthoc *risk of missed intent* in a conversation turn for the purpose of language model development, in which there is no existing literature.

To improve the language models, human-to-computer interactions need to be continuously reviewed. In our existing process, semi experts in a domain are given a random sample of recent chats collected from a live system for review. They manually grade them in an effort to find intents which need improvement. These reviewers need only be familiar with any domain specific terminology to be qualified to review chats. If a reviewer decides that an intent was not appropriate given the user input, they may recommend a different intent. The result of the reviewing process is a set of chats and their grades, which are passed to the domain experts. Only poorly graded chats are analysed by domain experts to make the necessary changes to the language models.

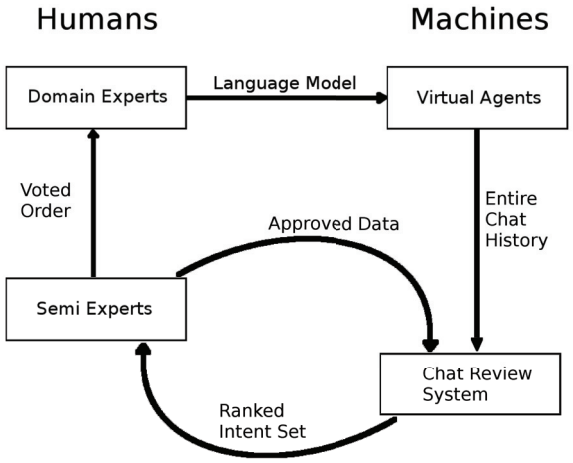

Figure 1: Augmented Language Model Development Cycle

As this is a manual and time consuming process, the reviewers are only able to view a limited number of chats. The result is also subjective since reviewers may disagree on the appropriate intent for a chat. In addition, as the language model improves, miscategorized chats become more difficult to identify in a random sample due to their dwindling numbers.

To increase the efficiency and coverage of the development process, we created the Chat Review System (CRS). The CRS augments the review cycle by preprocessing the entire chat history from a virtual agent, and prioritizing chats for the reviewers. It ranks chats by risk of missed intent, which we define as *riskiness*. The individual chat risk score is derived from the presence of one or more risk indicators which will be described in detail below.

The CRS is placed between the virtual agents and the reviewers as shown in Figure 1. Reviewers are presented with the riskiest chats first, and they vote on if a given chat was

assigned to the best intent, in place of grading. The CRS then determines actions to be taken on chats and intents based on riskiness and reviewer feedback. The chats that are determined to be correctly mapped by the language model after review are fed back into the CRS as training data. The set of chats, associated voting outcomes, and recommended actions are then given to the domain experts for detailed guidance on language model improvements.

In the following sections we describe the CRS architecture, how the chat risk score is derived from risk indicators, development process improvements, and performance measurements.

Chat Review System

Since a large number of reviewers may not be on-site, the CRS was designed as a web application. Each language domain is isolated in the CRS as a project, which may contain multiple deployments of a language model. The language models under review were previously deployed to a live virtual agent which may be exposed to human users over channels such as a website, Short Message Service, Interactive Voice Response, or native mobile applications. Each deployment is given a unique version number that allows the CRS to evaluate chats from the live system against the exact language model used.

Dataset Creation

When a language model is to be reviewed, it is uploaded to a project along with the entire live chat history using that language model version and the set of regression chats.

Regression chats are used by domain experts to verify language model behaviour during construction and refinement. Each intent has a set of chats assigned by a domain expert as an example of language that maps to it. These regression chats may be constructed by the designers or come from a live system and saved after review. The number of regression chats varies greatly by maturity of language model, but more mature sets may have hundreds per intent.

The versioned language model and chat history pair are referred to as a *dataset*. The live chats are exported from a virtual agent's reporting database into a dataset. These exports retain the user input, intent hit, conversation identifier, and order in conversation. Optionally, they may include any user survey results, if collected. A real conversation from a live virtual agent is given in Table 1.

Aggregating data from all deployed virtual agents requires the CRS datastore and queries to scale horizontally as even a single agent may see several million chats in a month. MongoDB was chosen for these properties (Horowitz 2011) and its built in MapReduce framework which the CRS uses heavily. It has been shown to handle aggregating large volume chat data and subsequent querying well (Beaver and Dumoulin 2014).

Chat Analysis Overview

The CRS performs several passes of risk analysis. In each pass, indicators of risk discovered are added to a list associated to the individual chat. Each recurrence of an indicator is added to the list. Specific indicators of risk and how they are combined will be covered in a following section. In the first pass, the CRS looks at conversation level features. These are indications of risk in the structure of the conversation. The second pass looks at chat level features, which are indications of risk at the individual chat level.

Chat Text	Intent Hit	Conversation ID	Entry #
My TV is not working I need to have it fixed.	TV Support	26789	1
My TV will not start I want to talk with support staff	Contact Information Deflection	26789	2
I cannot get my tv to work	TV Support	26789	3
How can I get a person to help me get the TV started?	TV Services	26789	4
How can I speak with a support staff to get my TV to work?	TV Support	26789	5
There are no links showing my problem what now?	I Don't Know	26789	6

Table 1: Conversation with Risky Inputs

In parallel to this process, the CRS builds classifiers for use in the analysis. Any supervised learning method that supports multi-class classification may be used, and multiple classification methods are used if present. When using multiple methods, each classifier is trained on the same training data. This training step is done in parallel so that the training period is only as long as the slowest training interval. By training multiple classification methods from the same data, we theorize that any disagreement in the highest ranked intent between a single classifier and the language model is an indicator of risk. The CRS then runs chats through each available classifier and compares its highest ranked intent to the one originally selected by the language model.

After the classification pass, the risk indicators present are combined into a singular risk score for each chat. The scores are then normalized to obtain a ranking from highest to lowest risk within the dataset.

Voting Process

The chats are selected for review in order of highest risk first, and a threshold is set for the number of chats to be reviewed. This threshold is configured by the domain experts and can be changed at any time if they feel they need more data for their investigation. The reviewers are asked if a chat belongs to the intent it was mapped to, and are given the possible responses of *Yes*, *No*, or *Not Sure*. They are presented supplementary information to inform their choice; including the full conversation in which the chat appeared, the regression chats assigned to that intent for comparison, and a list of intents that are related by similarity of input language to the current one. To control for the subjective nature of voting, each chat is presented to a minimum of three reviewers. The majority vote is calculated and used as the basis for missed intent.

Risk Indicators

The categories of risk that we attempt to detect and the risk indicators within them are defined in Table 2. Each risk indi-

I Don't Know (IDK) response risk indicators	
input_triggers_idk	This input directly triggered an IDK response
input_preceeds_idk	The input directly preceded an interaction that included an IDK response
idk_in_conversation	The input was involved in a conversation that contained an IDK response
input_triggers_impasse	This input directly triggered an impasse
Instances of multiple hits risk indicators	
input_triggers_seq	This input triggers a response that is directly involved in a sequential hit
seq_in_conversation	The input is involved in a conversation that contained a sequential hit
mult_in_conversation	The input is involved in a conversation that contained multiple hits to the same intent (but were not sequential)
User feedback score risk indicators	
usr_score_1	Rating score of 1
usr_score_2	Rating score of 2
usr_score_3	Rating score of 3
usr_score_4	Rating score of 4
usr_score_5	Rating score of 5
Classifier agreement	
ci_predictor:	Probability of not being the original intent
pni_origin:	Penalty given to the intent a Potential New Input (PNI) originated from

Table 2: Risk Indicators

cator is given a weight $w \in [0, 1]$, initially 1, that determines the impact the indicator has on the overall risk of a chat. The selection of these indicators was done by conducting interviews with experienced reviewers on attributes they look for in chats that are graded poorly. For example, in Table 1, it is easy for a human to see that the user is not getting a satisfactory response from the virtual agent. In order to train a system to reach the same conclusion, we attempt to extract from human reviewers what features of such a conversation they observe.

We consulted domain experts for which of the proposed indicators from the interviews were of the most importance to detect language model errors. These indicators in Table 2 reflect the tribal knowledge of our existing reviewer base. Other indicators may exist, and this is a subject of future research.

I Don't Know (IDK) response risk indicators The first category in Table 2 applies to what we call "I Don't Know" (IDK) responses. These occur when the language model does not find an intent that satisfies the input. An IDK intent may return a response such as "I'm sorry, I don't understand you. Please revise your question.". An example of this is seen at entry 6 in Table 1. Therefore all chats in that conversation would be tagged with *idk_in_conversation*, entry 6 would be additionally tagged with *input_triggers_idk*, and entry 5 with *input_preceeds_idk*.

A language model may also contain intents that are used as "intelligent" IDKs. An example response to one of these may be: "I see that you are asking about liability insurance, but I do not have detailed knowledge in that area.".

Another type of IDK occurs when the same intent is hit more than two successive times within a conversation. This is an *impasse*, since it is clear we cannot give the user a satisfactory response. It may indicate that the input mapping associated to the impasse intent is too broad, or the virtual agent is missing domain knowledge.

Instances of multiple hits risk indicators The second category applies to multiple hits, where the same intent was returned multiple times in a conversation. This is an indication of risk within a customer service conversation as it is unlikely the user would want to see a specific response more than once. An example of this are entries 1, 3, and 5 in Table 1. If two hits are successive, we label the interactions as *sequential hits*. This usually indicates that the response to the first input did not satisfy the user so they are rewording their questions to get a different response.

User feedback score risk indicators The third category is derived from ratings given by the human users themselves. Typically, customer service interactions present surveys on user satisfaction after a conversation is complete. If these ratings are present, they can be used as an indicator of risk. If the user reported he/she was unhappy with the conversation, it may indicate the intents present within that conversation need improvement. The ratings are normalized into a range [1,5] where 1 would represent a poor response and 5 would represent an excellent one. Each rating score is given its own risk indicator in order to weigh them individually.

Classifiers agreement The final category is used by classifier models to indicate risk based on the class score of the intent that was assigned by the language model. As described previously, all chats are ran through classifiers, and the chats are tagged with a *ci_predictor* indicator per classifier, which carries a weight derived from the probability of class membership.

For simplicity's sake, we consider a single classifier. We use Scikit Learn's Support Vector Classifier (SVC) with a linear kernel and $C = .5$ as our classification method. It implements a "one-vs-rest" multi-class strategy. To determine a weight for risk ci_predictor, we use SciKit Learn's decision_function method which returns the signed distance of a sample to the hyperplane. Using this distance d, we can calculate the probability P of class membership. We use the following estimation method since datasets may have input counts in the millions and calculations must be performed on each input:

$$P = \frac{d}{2} + .5 \tag{1}$$

As we need an estimation method that will perform at a consistent and efficient speed with a large number inputs,

and we do not require a high degree of precision, we use the estimation technique in (1) over other techniques such as Platt scaling (Platt 1999). Platt scaling has been shown to be an expensive operation on large datasets (Chang and Lin 2011). Note that if d does not satisfy

$$\epsilon \leq \frac{d}{2} + .5 \leq 1 - \epsilon \tag{2}$$

where $\epsilon = .0001$ in our case, then P will take on the value ϵ or $1 - \epsilon$, whichever is closer.

Let $1 - P$ be the value of ci_predictor for the assigned intent. If a classifier returns a high P, the assumed risk of missed intent is low.

A second way classifiers are used in the CRS is to help predict which intent is correct, given the classifier does not agree with the original mapping. If a classifier returns a different intent ranked highest, and it has a confidence in its choice that is greater than 0.6, the CRS generates a new input associated to its top ranked intent with the same text. These are named Potential New Inputs (PNIs), and will be voted on along with the original intent mapping. The original chat from the live system will be tagged with a *pni_origin* indicator. The threshold of 0.6 was arbitrarily chosen, and one of the future directions for this research will be to optimize it.

If an intent has a large number of chats tagged with a pni_origin indicator, it is evident that the intent is consuming too broad of language and needs investigation. If an intent has a large number of PNIs assigned to it, the evidence suggests that it has too narrow of language or is redundant and also needs investigation.

Chat Risk Score

The score for risk of missed intent is computed from the risk indicators present in a chat as follows. Let C be the set consisting of all input chats. Assume that a chat $c \in C$ has n risk indicators present. Let w_j represent the weight for risk indicator j. The risk score for chat c, known as z_c, is defined as:

$$z_c = \sum_{j=1}^{j=n} w_j \tag{3}$$

We then normalize the chat risk scores across the dataset as follows. Let $MaxScore$ be the maximum chat risk score. The normalized chat risk score is defined by:

$$z'_c = \frac{z_c}{MaxScore} * 100 \tag{4}$$

This normalized value, z'_c, is assigned to the chat as its risk score. It is its measure of risk relative to the riskiest chat in the dataset.

CRS Output

After the risk analysis and voting stages have been completed, the CRS exports the voting data and additional recommendations to the domain experts to facilitate language model development. The CRS flags an intent to be reviewed in the following circumstances:

- A chat-to-intent map is voted to be incorrect
- The majority votes *Not Sure*
- There is no majority consensus
- A PNI is voted to belong to the suggested intent

There is a situation when a PNI is generated and a majority agrees the chat belongs to both intents. These chats are surfaced by the CRS, and the domain experts can choose to re-release them for voting. If not, the CRS will flag both intents for review. If a chat-to-intent map was voted to be correct, no action is needed. As seen in Figure 1, the voting data is fed back to the CRS as training data for future classifiers, as well as weight optimization.

Evaluation Dataset Creation

To measure the performance of the CRS we constructed three datasets, each from a different language domain. The CRS must work well in any domain so we are motivated to train weights in a domain agnostic way. Each dataset was generated from a random sample of approximately 150 full conversations taken from a live virtual agent. All chats were selected for voting and released to a group of 15 reviewers. After voting, the average number of chats per dataset with a clear majority was 860. If there was no clear majority, the chat was not used as an evaluation chat.

The language models for these datasets differ in their *maturity*. The maturity of a language model is dependent on its length of refinement. As we have virtual agents with various refinement time, we created three categories and selected a domain from each:

Dataset	Refinement Time	Domain	# Chats
1	0-12 months	Airline Service	921
2	12-36 months	Retail Support	871
3	36+ months	WebForm Help	788

To account for maturity differences, Dataset 1 and 3 were used for training risk indicator weights, and Dataset 2 was used for CRS evaluation. Analysis was limited to a small number of conversations as the construction of a gold standard with multiple reviews per chat is a very expensive undertaking. However, the CRS was designed to scale and has processed tens of millions of chats in production use.

Optimization of Risk Indicator Weights

Weights associated with each risk indicator are optimized using the training datasets. Chats marked as incorrect by a majority of voters are pushed to the top of the review priority list as a result of their risk score. The following sections describe the process of discovering which risk indicators are the most influential in determining risky chats.

Determination of Weights Using ORs

Odds Ratios (ORs) are commonly used in many fields of social science and medical research to compare the impact of risk factors on a selected outcome [(Parrish et al. 2011), (Etchegaray et al. 2012)]. The odds ratio represents the odds that an outcome will occur given a particular exposure, compared to the odds of the outcome occurring in the absence of that exposure (Szumilas 2010).

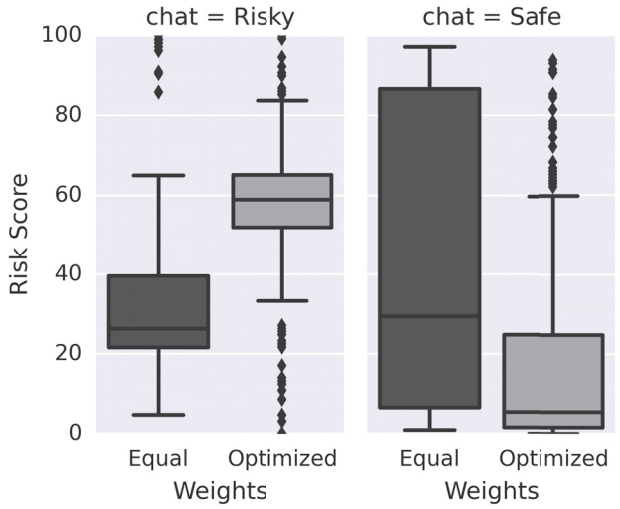

Figure 2: Comparison of Risk Scores

From our voter evaluated datasets, odds ratios for every risk indicator are calculated. Statistically insignificant risk indicators are eliminated using a 95% confidence interval. Remaining indicators are then ranked by their ORs where a higher OR indicates a larger magnitude of effect. Finally, ORs are normalized between 0 and 1 to obtain weights for their respective indicators.

To calculate ORs, dichotomized exposures must be delimited. An input chat is *risky* if the majority of voters disagreed with the intent it was mapped to, otherwise it is labelled as *safe*.

$$\begin{array}{c} \\ \text{Risk Indicator} \\ \text{No Risk Indicator} \end{array} \begin{array}{c} \begin{array}{cc} \text{Risky} & \text{Safe} \end{array} \\ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \end{array}$$

$$OR = \frac{a/c}{b/d} \tag{5}$$

Since ci_predictor is a probability returned from a classifier, it is a continuous variable from 0 to 1. To dichotomize ci_predictor, we perform the following: If the classifier and language model agree on the top intent, ci_predictor will not be considered present. If they disagree, ci_predictor is present. The value of the OR indicates the effect the risk indicator has on the riskiness of the chat: an $OR \gtrsim 1$ signifies that the indicator positively affects riskiness. If the 95% confidence interval of the OR includes 1, we deem it a statistically insignificant result. The lower and upper bounds for the 95% confidence interval are calculated in the following manner (Szumilas 2010) :

$$e^{ln(OR) \pm 1.96 * \sqrt{(\frac{1}{a}+\frac{1}{b}+\frac{1}{c}+\frac{1}{d})}} \tag{6}$$

Finally, weights are given to risk indicators by normalizing the ORs.

If risk indicators are not independent, an adjusted OR can be calculated with logistic regression (Santos 1999). We observed that risk indicators tend to occur individually without the presence of other risk indicators in our datasets. Thus, unadjusted ORs are used.

Result of Optimizations

Two evaluated datasets are considered for weight optimization. We test these obtained weights on a third dataset. Equal weights are initially set for each risk indicator. An acceptable input for the determination of ORs must have at least three voters and a majority vote of either *Yes* or *No*.

The ORs and confidence intervals are evaluated for the risk indicators determined by Dataset 1. Insignificant risk indicators are removed, and weights of risk indicators are obtained by normalizing the ORs:

Risk Indicator	Weight
input_triggers_idk	1.0
ci_predictor	0.4184
usr_score_1	0.0871
input_preceeds_idk	0.0546
idk_in_conversation	0.0475
usr_score_2	0.0395
input_triggers_seq	0.0279
usr_score_4	0.0021
usr_score_5	0.0

This process is repeated for Dataset 3:

Risk Indicator	Weight
usr_score_2	1.0
idk_in_conversation	0.5841
ci_predictor	0.4917
input_triggers_idk	0.3981
input_preceeds_idk	0.2000
input_triggers_seq	0.0

No weights are obtained for *Mult_in_conversation, seq_in_conversation, input_triggers_impasse,* and *usr_score_3* due to their insignificance in both Datasets 1 and 3. If a risk indicator weight only exists in one dataset, it is the weight that is used. Otherwise, we average the weights. Averaged weights from Dataset 1 and 3 follow:

Risk Indicator	Averaged Weight
input_triggers_idk	0.7341
usr_score_2	0.4639
ci_predictor	0.4508
idk_in_conversation	0.2846
input_preceeds_idk	0.1188
usr_score_1	0.0871
input_triggers_seq	0.0156
usr_score_4	0.0021
usr_score_5	0.0

These averaged weights are applied to our test set. Performance with equal weights is compared to performance with optimized weights using the chat risk score, z_c', in (4).

If optimized weights are working appropriately, z_c' should be higher for chats labelled as *risky* and lower for chats labelled as *safe*. This exact behavior is displayed in Figure 2 where the median risk score for *risky* chats shifts up as a direct result of our implementation. An even more dramatic change is evidenced with *safe* chats, where the risk score plummets. With equal weights, the distribution of risk scores for *safe* chats is rather large. Optimized weights tighten the distribution making z_c' a more valuable metric.

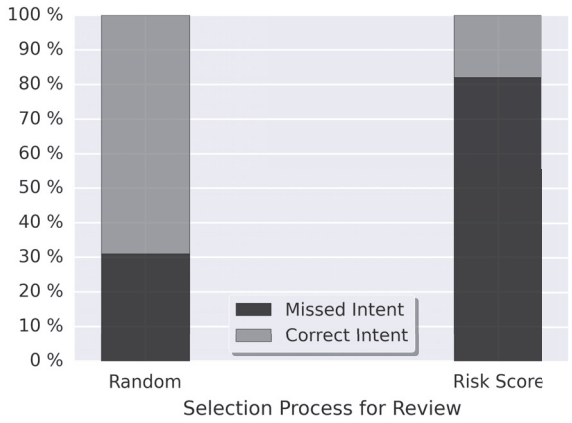

Figure 3: Comparison of Review Selection Processes

CRS Performance

Our primary measure of CRS performance is the prioritization of chats with missed intent. Reviewers are given a fixed number of chats to review due to time constraints; therefore, maximizing the number of chats with missed intent in a fixed sample size is our goal. To do this, Dataset 2 was processed by the CRS using the optimized indicator weights previously determined. There are 278 chats with missed intent out of the 871 in Dataset 2, or 32%. For evaluation of our methods, 200 chats were selected using the original random process, and 200 were selected by top risk scores from the CRS.

Random selection is used as there is no literature on this specific topic to compare against. For the random process, correct intent and missed intent counts were averaged over 3 independent selections, using Python's `random.randint` function to select chat IDs from the database.

As Figure 3 demonstrates, the percent of missed intents found in the selection increased from 31.5% to 82% using the CRS risk score in place of a random sample. This is a very significant increase in productivity for the reviewers, as the majority of their work is now directed to chats that legitimately need review.

Through further research on features of risky chats, our goal is to replace human voters altogether as risk detection improves, thereby allowing the CRS to process the chats in real time without the delay of the voting process. When to replace the voters is subjective, as it is dependent on the tolerances of the domain experts for false positives; research is ongoing into when voting can be safely removed and how domains affect risk thresholds.

Conclusions

We have presented a system for autonomously evaluating chat data and prioritizing language model improvements for chatbots, regardless of their implementation. We introduced indicators of risk and a method to weight them. This system alleviates the subjective nature of individual human reviewers. All of the data is processed but only the riskiest is presented to humans for review. With weighted risk indicators, we demonstrate a significant improvement in locating chats with missed intent in a fixed sample size. The subjective nature of human suggestions for correct intent is removed, as the CRS generates PNIs only if confident, and verifies them with a voter majority. Thus, language model development is improved by surfacing chats with missed intent and providing reviewed suggestions to guide repair.

References

Beaver, I., and Dumoulin, J. 2014. Scaling user preference learning in near real-time to large datasets. In *Modern Artificial Intelligence and Cognitive Science Conference 2014*, volume 1144, 68–73. CEUR-WS.org.

Chang, C.-C., and Lin, C.-J. 2011. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2(3):27.

Cohen, P. R.; Morgan, J. L.; and Pollack, M. E. 1990. *Intentions in communication*. MIT press.

Dumoulin, J. 2014. Using multiple classifiers to improve intent recognition in human chats. In Inoue, A., and DePalma, P., eds., *Modern Artificial Intelligence and Cognitive Science Conference 2014*, number 1144 in CEUR Workshop Proceedings, 10–21.

Etchegaray, J. M.; Ottenbacher, A. J.; Sittig, F.; and McCoy, A. B. 2012. Understanding evidence-based research methods: Survey analysis, t-tests, and odds ratios. *HERD: Health Environments Research & Design Journal* 6(1):143–147.

Holtgraves, T. 2008. Automatic intention recognition in conversation processing. *Journal of Memory and Language* 58(3):627–645.

Horowitz, E. 2011. The secret sauce of sharding. MongoSF. Available online at http://www.mongodb.com/presentations/secret-sauce-sharding.

Montero, C. A., and Araki, K. 2005. Enhancing computer chat: Toward a smooth user-computer interaction. In *Knowledge-Based Intelligent Information and Engineering Systems*, 918–924. Springer.

Parrish, J. W.; Young, M. B.; Perham-Hester, K. A.; and Gessner, B. D. 2011. Identifying risk factors for child maltreatment in alaska: A population-based approach. *American Journal of preventive medicine* 40(6):666–673.

Platt, J. C. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*. Citeseer.

Pradhan, S. S., and Ward, W. H. 2002. Estimating semantic confidence for spoken dialogue systems. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, I–233. IEEE.

Santos, S. I. 1999. *Cancer epidemiology: principles and methods*. IARC.

Szumilas, M. 2010. Explaining odds ratios. *Journal of the Canadian Academy of Child and Adolescent Psychiatry* 19(3):227.

Wessel, F.; Schlüter, R.; Macherey, K.; and Ney, H. 2001. Confidence measures for large vocabulary continuous speech recognition. *Speech and Audio Processing, IEEE Transactions on* 9(3):288–298.