Project Background

The User Intent classifier seen in my final project began last year as an open-ended exploration of my company's (Ask.com) keyword portfolio. Previously, these keywords had been examined for the purpose of building thematic classifiers that recognized keywords were about specific topics (such as "Health" or "Sports), and had also been segmented in various ways based on user interaction in order to assess their value. This "deep dive" (as it was termed) was initiated to gain a deeper knowledge of the character and behavior of keywords in the portfolio.

Based on my initial analysis of the portfolio keywords, I developed a set of User Intent classes, helped select a segment of the keyword portfolio for labeling, and created labeling guidelines for an outside vendor, Crowdsource. Using my guidelines, Crowdsource performed a larger-scale labeling task of ~100k Health-related keywords for User Intent. This data set is used in my final project in an attempt to create a user intent classification model.

Initial Exploratory Analysis

For my initial analysis, I looked at segments of the keyword portfolio, including:

- Top Performing Keywords (Based on clicks and traffic)
- Samples of our top performing thematic categories: Health, Home&Garden, Vehicles, Travel
- Random Cross-Categorical Samples

These samples were chosen to represent the overall breadth of the portfolio, but we also wanted to give special consideration to the keywords that users engaged with most frequently (and therefore the keywords with the greatest impact on our company).

The initial analysis was pretty rough-- I manually skimmed through large samples of the keyword portfolio and used regular expressions to extract patterns that I found in the keywords. Based on these patterns, I proposed a set of User Intent classes for the keywords that would classify the keywords from the user's perspective. The classes would attempt to answer the question: "What does the user want to obtain or hope to accomplish when typing in this keyword?" The details of these User Intent classes are included in the "Data Set Description" section below.

We did some in-house hand labeling of keyword samples using these classes, and found that there is an interesting relationship between user intent and subject category. We had a few key insights:

- User intent classes can have a very different structure across subject categories. Meaning, a Resource keyword in Health could look very different from a Resource keyword in Vehicles:
    - Health: charts, graphs
    - Vehicles: manuals, diagram
- Subject categories often have an overall "theme" and feature one or more major user intent classes
    - Vehicles: Mostly Transactional and Resource
    - Health: Mostly informational classes (Generic and Direct Answer)

Data Set Selection

Based on conversations with product stakeholders and some experimental labeling over various thematic categories of the keyword portfolio, we decided to use a sample of keywords that were classified as Health by our in-house thematic classification system. The Health category has a strong definition and is thematically consistent-- it is almost certain that all keywords categorized as Health by the thematic classification system are related to health and wellness. These keywords are also mostly unambiguous-- we would be less likely to run into labeling errors that might occur in other categories. It also seemed logical to focus our efforts to understand user intent on a category with high volume and that contains keywords that exhibit a high degree of user engagement.

Data Set Description

The data used for this User Intent classifier consists of 90,835 keywords from the keyword portfolio that were classified as Health by our in-house thematic classifier. These keywords were then labeled for User Intent by Crowdsource, based on the classes described below:

- **Navigational**
    - User wants to navigate to a website, or information hosted on a specific website. Business names and government agencies without additional context are included in navigational.
    - Example: aetna login, myhealth com
- **Resource**
    - User wants to find a specific type of resource or media type. The resource can be confined to the web, such as reviews, calculators, or look ups, OR the resource can be something that can be printed out, downloaded, or viewed.
    - Example: diabetes blood chart, pictures of skin rash
- **Generic**
    - The keyword has a clear topic, but no additional context which would indicate what the users would like to know or see. Generic seems to imply the user request: "Show/Tell me *anything* about Topic"
    - Example: breast cancer, facts about cellulite
- **Direct Answer**
    - The keyword indicates that the user has a specific question, and is looking for a specific answer. The length of the answer can vary from a single word to an entire article. This label includes requests for lists, or keywords that would prompt a list-like response. Direct Answer implies the user request: "Show/Tell me *X* about Topic". Example: list of flu symptoms, what is the difference between diabetes I and II
- **Guides & Instructions**
    - User's intent is to find the steps to accomplish a task or project. This includes instructions, guides, and recipes.

    - Example: how to treat wasp sting, change wound dressing
- **Transactional (Shopping Intent, or Product Name)**
    - This category can be applied to keywords with any explicit transactional component--keywords about buying, selling, renting, prices, costs, values, etc. It also is applied to all product names, any item that can be bought or sold. For this specific use case, it made sense to include product names along with explicit shopping requests.
    - Example: truvada cost, buy viagra

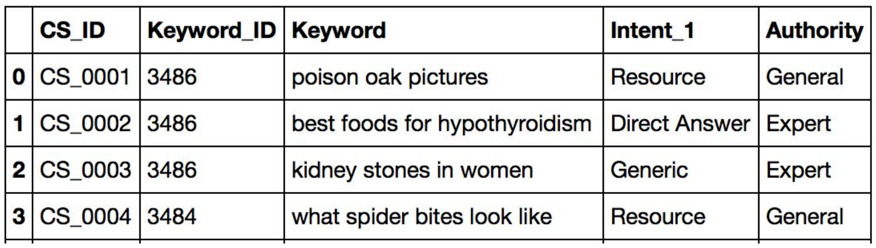
The data also includes another labeled class called **Authority.** For this label, we asked the Crowdsource evaluator to use their personal judgment to select the level of expertise they would want or expect from a responder to a given keyword. I also asked labelers to add a note if the keyword was **Temporal** (time-dependent) or **Local** (included a location name or required insight into the user's location in order to fulfill the request)**.**

Problem Statement

My goal for this experiment is to see whether it's possible for me to create a User Intent classifier that performs reasonably well over keyword data. This project almost feels like more of a test of the quality of the User Intent classes that I created for the original labeling initiative--are they coherent and mutually exclusive enough for use in a machine learning model? Where will the classes overlap, or conflict with each other?

Labeled Data Set Preprocessing

I am very fortunate that while a great deal of work had to go into the initial analysis and project definition, the data set itself is remarkably clean.



	CS_ID	Keyword_ID	Keyword	Intent_1	Authority
0	CS_0001	3486	poison oak pictures	Resource	General
1	CS_0002	3486	best foods for hypothyroidism	Direct Answer	Expert
2	CS_0003	3486	kidney stones in women	Generic	Expert
3	CS_0004	3484	what spider bites look like	Resource	General

The columns CS_ID, Keyword_ID, and Authority were dropped, as they will not be used in the classification model. I also converted the User Intent labels to integer values.

I created a small function to tidy up the keywords by lowercasing them and stripping out non-letters (numerals, misc. characters).

At this step, I also experimented with several approaches to stopword lists, in an attempt to pull out English function words that occur frequently in the documents, but often do not contribute meaning (i.e. articles such as "a" and "the").

I experimented with several approaches to stopwords: NLTK's built in stopword list, MySQL's stopword list, and playing with the max_df and min_df parameters of CountVectorizer. All of these stopword experiments caused my model to decline in performance. The results of these experiments confirmed some observations from my initial analysis of user intent classes-- for some user intents, the inclusion of certain terms is a key part of the classification, while for others, the underlying structure of the keyword provides a strong signal to its classification. For example, the most common terms in the Resource category, as compared to the most common terms in the Direct Answer category:

Resource: images, photos, rash, skin, cancer, chart, picture, form, list

Direct Answer: causes, what, effects, side, is, cancer, pain, signs, how

In this example, two things stand out to me: 1) Resource keywords have an obvious semantic pattern-- similar object types feature highly in the category, 2) Direct Answer seems to have a subtle syntactic patterns-- many of the most common tokens are plural, and several function words (the question words 'What' and 'How' and the auxiliary 'is') also appear frequently in the category. This is an indication that Direct Answer keywords pivot on syntactic structure, while Resource keywords pivot on common objects.

After more investigation, it became clear that syntactic structure (question-formed syntax and plurality, for example) were important to the Guide and Direct Answer categories, while meaningful tokens were important to the Navigational and Direct Answer categories. Ultimately, I ended creating an extremely constrained stopword list that only contained English function words that did not negate these important syntactic patterns.

Vectorization

In order for my keywords to be used in a model, they needed to be transformed into feature vectors. I used Sci-Kit's CountVectorizer to convert the keywords into a matrix of feature counts. Although token count can be really useful, it's not exactly the best indicator that a given token is important-- some words might just appear more commonly than others. Thus, I implemented Sci-Kit's TfidfTransformer. Tf-Idf stands for "Term Frequency times Inverse Document Frequency". With Tf-Idf, the importance of a token increases proportionally to the number of times a word appears in the document ("term frequency"), but is offset by the frequency of the word in the corpus "inverse document frequency".

I implemented a fairly comprehensive GridSearch to determine which parameters would yield the most successful feature set for my model. Amusingly enough, the GridSearch determined that my model would be happiest with the default settings for CountVectorizer and TfidfTransformer. I'm curious to play with these features a little more to see if I can't improve performance somewhat.

Model Selection

Before building my model, I had a conversation with Justin during office hours and he suggested that I first build a binary model that would tackle the most prominent User Intent classes in my data-- Generic and Direct Answer. After building that model, I would pipe the output to a second model that would handle the smaller, less representative intents: Resource, Guide, Navigational, and Transactional.

I tried several algorithms for both the binary and the multiclass classifier before selecting my final model. The models I tried were:

- Naive Bayes
- LinearSVC
- SGDClassifier

I ran all three over my data and got the following results:

Results of LinearSVC

	Predicted Class Generic	Predicted Class Direct Answer
Actual Class Generic	7559	1034
Actual Class Direct Answer	1704	5640

Precision: 0.845070422535
Recall: 0.767973856209

Results of SGDClassifier

	Predicted Class Generic	Predicted Class Direct Answer
Actual Class Generic	7872	721
Actual Class Direct Answer	1953	5391

Precision: 0.882035340314
Recall: 0.734068627451

Results of Naive Bayes

	Predicted Class Generic	Predicted Class Direct Answer
Actual Class Generic	6847	1746
Actual Class Direct Answer	1849	5495

Precision: 0.758873083828
Recall: 0.748229847495

In retrospect, I would have tried to find the optimal parameters for each model and then run them over my data, however, at this point in the class we had not covered grid search. I was impressed by the performance of LinearSVC and SGDClassifier. While LinearSVC appeared to have slightly better recall performance, SGDClassifier had higher baseline precision. Also, after reading documentation on the two options, I learned that SGDClassifier requires less memory and has some additional options for regularization.

Model Results & Validation

My final model ended up being two SGDClassifiers run in parallel, rather than piping the output of one classifier to the second. When determining the optimal parameters for my classifier using a grid search, it turned out that the Generic/Direct Answer data and the Other Intents data had different requirements. Here are the results:

Generic/Direct Answer Model

Class Name	Precision	Recall	F1-Score	Support
Generic	.8	.92	.86	10764
Direct Answer	.88	.74	.8	9157
Avg. Total	.84	.83	.83	19921

Accuracy Score: 0.832653573445
AUC: 0.825558141006
Cross-Validated Accuracy: 0.828346165856 (cv=5)

Other Intents Model

Class Name	Precision	Recall	F1-Score	Support
Guide	0.98	0.98	0.98	2197
Navigational	0.82	0.94	0.88	299
Resource	0.98	0.96	0.97	1800
Transactional	0.97	0.94	0.95	318
Avg/Total	0.97	0.97	0.97	4614

Accuracy Score: 0.832653573445
AUC: 0.825558141006
Cross-Validated Accuracy: 0.828346165856 (cv=5)

Challenges

Unbalanced Classes
Distribution of Intents:

Intent Label	Raw Count	Percent of Sample
Generic	43151	43%
Direct Answer	37765	38%
Guide	9138	9%
Resource	7293	7%
Transactional	1238	1%
Navigational	1196	1%

I was not surprised by this distribution of intents-- as mentioned, in the initial exploratory labeling for this project, we discovered that subject category has an impact on the types of user intent classes most frequently represented in the data. Since my classes were so radically imbalanced, that raised some concerns for me in terms of my model.

I didn't want to undersample or oversample Generic and Direct Answer, since these intents had the most complex and varied syntax structure and vocabulary, so those were split out into a single binary model (which I was going to do anyway). For the remaining intents, I decided to undersample to balance the classes, which improved the accuracy of the model. However, I found that SGDClassifier also has a parameter "class_weight" that, when set to "balanced" will balance the classes for you.

Labeler Error

Here is a confusion matrix for the binary Generic/Direct Answer classifier (test size=0.2):

	Actual Generic	**Actual Direct Answer**
Predicted Generic	7872	721

Predicted Direct Answer	1946	5398

Direct Answer has somewhat of a precision problem, so I created a function that would output a file of the model errors. So, I created a function that would output a file of errors. My team does a lot of work sifting through model error output and trying to identify patterns that can be addressed in order improve model performance. I wanted to figure out if there was anything wrong with my pre-processing steps or parameter selection that was leading to confusion between the two classes.

The error file contained output of instances where the model predicted one class, and the label in the data classified the keyword as belonging to the other class. While examining the error output, I found a problematic pattern: keywords formed "what is/are x". Around ~450 total errors contained the bigrams "what is" or "what are". It appears that labelers were confused by keywords with the construction "what is brain cancer" or "what are infectious diseases). There wasn't a consensus as to which class these keywords belonged to, so a proportion was labeled Generic and a proportion was labeled Direct Answer, leading to confusion.

The error output for the multiclass classifier had a similar story-- while the classifier had good performance, I was curious about a certain distribution of errors from the confusion matrix-- there seemed to be a higher number of errors between Resource and a few other classes (mainly Navigational and Guide). Looking back at the labeling guidelines sent to CrowdSource, I realized that the definition for Resource opened up the possibility for some labeler errors, as they created the potential for overlap between these categories.

Visualizations

As you notice, this paper is really boring-- there's a bunch of tables and numbers, but no interesting visual accompaniment that helps explain the patterns and phenomena that I saw in my data. This was a huge challenge for me, and an area where I desperately need to improve. I ended up omitting a portion of this report where I described my early explorations of the labeled data set, where I looked at the most common token n-grams in the data and distribution of certain syntactic patterns because I couldn't figure out a way to represent the information visually and had very little time to figure out how to do that. I did research visualization options for language data, but ultimately ran out of time. I intend to go back to this paper in the future and recreate my report with descriptive visualizations.

Potential Business Applications and Next Steps

Having insight into user intent would be very useful at my company-- at an exploratory level, having a functional user intent classification system could allow us to analyze the intersection of user intent and category in our keyword portfolio, and help us segment that portfolio to be used on different domains. User intent classification could also be used to determine what kinds of products or partnerships would be valuable for a given domain. For example, we discovered that there's a healthy proportion of Guide and Resource keywords in the Home&Garden category. So, based on this analysis perhaps it'd be useful to partner with a company that provides high-quality video content. If anything, creating user intent classes has been useful because it's provided a common language for people to discuss types of keywords at our company-- in meetings, now when someone uses the term "Resource" to describe a set of keywords, everyone is aware of what kind of keywords are being discussed.

The next steps for this project and my model involve attempting a more in depth analysis of the labeled data and being more sophisticated about feature selection and engineering. Here's my plans:

User Intent Classes

- Experiment with clustering: The user intent classes were created using some cumbersome, manual efforts. While these classes led to decent results, it'd be interesting to apply clustering algorithms to see what patterns a machine learning approach would draw out.

Improve Feature Extraction/Feature Engineering

- Stemming/Lemmatization: Neither were implemented in this iteration and they could prove useful for a larger set of keywords.
- Experiment with Word2Vec/Gensim to gain insight into semantic relationships between tokens in the keywords

Model Improvements

- Implement confidence score: SGDClassifier has a method called decision function that predicts the confidence score of the input-- the likelihood that the input belongs to a given class. Implementing a confidence score interests me because it doesn't seem like user intent classes are always mutually exclusive-- for example, the keyword "where to buy an iphone 6" has a Transactional quality too it, but also is formed like a Direct Answer. Confidence scores could be an interesting way to get at where categories continue to overlap with each other.
- Remake with final user intent classes: Following this large scale labeling initiative, I reworked the user intent classes. Prior to the reworking, the user intent classes were overly focused on the type of product that could be surfaced in order to address the underlying request of the keyword.

  For example, Guide was a class specifically to address a request for step by step instructions, however, there really isn't any reason why Guide keywords aren't Direct Answer keywords-- the only reason Guide existed was to capture a potential presentation of the requested information. It'd be interesting to see how the new classes fare.

- Attempt with cross-categorical data