Alphanumeric Recognition Test Results

Jenni McKienzie, August 2, 2009

Introduction

Travelocity has a product that due to multiple resellers has confirmation numbers in a variety of alphanumeric patterns. The codes are regularly analyzed to optimize the grammar, weighting the expected patterns more heavily. However, the success rate on these still leaves lots of room for improvement, and the patterns change regularly. It is nearly impossible to keep up with them, but opening the grammar to accept more patterns would do more harm than good.

The codes vary from five to fifteen characters in length. Nearly 90% fall into five distinct categories, with either all numbers or numbers plus specific alphas. The other 10% are extremely varied, and the term "pattern" gets used loosely at times. One such pattern is the number 4 followed by a single alpha followed by 5 characters that may be alpha or numeric. The recognition is obviously not so great on such a pattern.

The quest, then, is twofold. First, to better capture that 10%. Second, to improve recognition on the other 90% of confirmation codes. IBM did some research on individual character recognition substitutions in 2001 and 2002 (http://drjim.0catch.com/alpha2-acc.pdf, http://drjim.0catch.com/alpha-acc.pdf), but what about strings of alphas in the better recognizers in use today?

Methods

The decision was made to run a test against Travelocity's speech recognition engine with 6-character strings. The strings were generated using a Latin Square to get every character next to every other character. Then a set were added that doubled each character at some point in the string. This resulted in 288 6-character strings.

Each participant was asked to read 8 strings, so it took 36 participants to get one set of data. VocaLabs provided the participants. A little over 500 participants called in, and enough recordings were listened to and transcribed sequentially to get 10 full sets of data.

There were some hints given in the instructions. Each data set had a key associated with it (an animal name) that would trigger specific instructions about confusion between 0 and O, 1 and I. For example, for the string "3J2K1L," the IVR said, "Now on this one, that's the number 1. Go ahead." Because the participants received their instructions via the web, it was impossible to control the font to ensure that there wouldn't be confusability.

The 8 strings generated from the Latin Square and the double character sequences were rearranged to put the double character as utterance number 6 and put all the ones with hints at the end of the list so that the early ones flowed more easily in the instructions.

During the transcription process, if a participant didn't give the utterance requested, it was thrown out. The test was for recognition, not instruction following or reading. There were also some turn-taking issues in the design that weren't discovered until it was too late to do anything about it. Measurements were taken of what was recognized when the expected utterance was given. Whenever an utterance (or lack thereof) was thrown out, the same string from another participant was used to get the 10 full sets of data. The first task was to find 10 instances of each of the 288 strings being correctly said by participants.

The engine was set to return up to 9 entries in the nbest list. For each of these 2880 utterances, the recognition logs were checked and the following items were recorded.

- The call ID to link the utterance back to the recordings and logs (this is often the same for the entire set of 8 utterances in a test set, but often not if utterances had to be thrown out as noted above)
- The result in nbest:1 or $$$$$$ if nothing was returned
- What spot the actual utterance was returned in, or 10 if it wasn't in the nbest list at all
- The confidence score of the correct return, or 0.00 if it wasn't in the list at all
- Nbest:2 if the utterance wasn't in the list at all

Here is one set of data.

- The call ID is the same for all of them, meaning the caller correctly gave all 8 strings.
- For the first utterance, the recognizer returned the correct string in the fourth spot with a confidence of 0.02.
- For the second utterance, the recognizer returned nothing.
- For the third utterance, the recognizer got it right in the first spot with a confidence of 0.40.
- For the sixth utterance, the recognizer returned nine strings, but none of them was correct. Nbest:2 is also noted.

Num	String	Call ID	Nbest:1	Correct nbest slot	Correct conf score	Nbest:2 for 10
1	A9B8C7	2795531	A9DHC7	4	0.02	
2	7D6E5F	2795531	$$$$$$			
3	F4G3H2	2795531	F4G3H2	1	0.40	
4	KZLYMX	2795531	KZLYMX	1	0.94	
5	XNWOVP	2795531	XNWOVT	2	0.33	
6	TAAJXP	2795531	TLAAJX	10	0.00	KLAAJX
7	PUQTRS	2795531	PUQKRS	2	0.01	
8	2I1J0K	2795531	2I1J0K	1	0.78	

After the transcription was complete, the results were tabulated to show what was returned for each character spoken. The raw numbers are not very telling since the no matches decrease the number of utterances with a return for certain characters, and the strings with double characters artificially increase the number of utterances for certain letters. Thus, percentages were calculated for each pair. Percentages were only calculated when something was returned. If an "easy" character was in a string with a couple of "hard" ones that led to a no match, it seemed unfair to influence the results of all the characters in the string.

Results

Character Pairs

The table with the percentages of character pairs uttered and returned is on the next page. The utterances go down the left side; the recognizer's return in nbest:1 is across the top. To make the data as readable as possible on a single page, the percentages are written as .25 rather than 0.25, in a very small font, on a page turned to landscape orientation.

Correct recognitions are down the diagonal in green. Misrecognitions are in increasingly darker shades of blue. Values of .00 mean that there were misrecognitions, but more decimal places would be needed to see it.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	0	1	2	3	4	5	6	7	8	9
A	.82		.00	.00	.01				.01	.00	.01	.00		.00	.00																				.13	
B	.01	.74	.02	.08	.09											.01	.00			.00		.02				.01										
C	.00	.00	.91		.03		.00		.00											.01		.02				.02									.00	
D		.07	.01	.81	.06		.01				.00				.00	.01		.00		.02	.00	.00														
E	.01	.02	.02	.01	.92	.00	.00			.00					.00					.00	.01								.00							
F						.69			.01			.02			.01			.00	.26		.00			.01	.00			.00								
G			.01	.01	.01		.93	.01		.01	.00					.00				.02						.00										
H	.00				.00			.98																											.01	
I									.96				.00		.00			.02							.00			.00				.00		.00		.00
J	.00						.02			.96	.01		.00								.00					.00							.00			
K	.01				.00					.02	.94		.00		.00	.00	.00			.01					.00											
L											.00	.97	.01	.00	.02				.00								.00									
M						.00			.00	.00		.01	.92	.06	.01						.00															
N									.00			.00	.04	.94					.00		.00															
O	.00							.00				.01			.97		.00									.00	.01			.01						
P		.02	.01	.01	.04	.00	.00				.00				.00	.74			.00	.15				.00					.00							
Q				.00	.00		.00								.01	.00	.82			.00	.12		.00				.00	.00	.04							
R				.00		.00			.01									.98													.01					
S						.09						.00	.00			.00	.00		.89		.00								.00							
T	.00	.00	.04	.00	.02		.04				.01					.06		.00	.00	.82		.00								.00					.00	
U	.00				.01										.00	.00	.00	.00			.98															
V		.08	.03	.04	.05		.01		.00						.00	.01					.00	.59				.18		.00		.01				.00		
W	.00			.00	.00	.00			.00	.00		.00			.00	.00	.00						.97													
X	.01				.00		.00	.00	.00	.00	.00								.03				.00	.91				.00					.02			
Y	.00						.00		.00						.00						.00			.00	.98		.00	.00				.00				
Z	.00	.00	.05	.00	.04	.00	.01			.00	.00			.00	.00				.00			.10		.00	.00	.76	.00			.00	.00			.00		
0						.00						.00			.01						.00	.00		.00		.00	.97	.00	.00		.00		.00			
1					.00				.00									.00							.07			.92	.00		.00					.00
2							.00					.00			.01	.00	.04			.01	.02						.00		.92					.00		
3			.01																	.00		.00				.00				.98					
4															.00			.01													.99					
5	.00			.00					.07									.00							.02						.00	.91				
6			.01		.00					.00	.00														.01	.01	.01			.00				.97		.00	
7									.00	.00				.00	.00										.00		.00		.00						.98	.00	
8	.04	.01	.00	.00	.02		.00	.16	.00																	.00									.00	.75	
9	.00								.01																.00		.00	.00								.98

The dark blue cells are the ones of obvious interest, but it's interesting to note that some characters had multiple incorrect returns at high rates, so the overall right/wrong numbers are interesting as well. Here are both sets of data. Overall right/wrong is shown both alphabetically and by how often that character is wrong. For that second listing, characters in blue are ones that are in the highest offenders list. What's interesting to note is that several characters that are often wrong (T, C, X, and 1) don't have any single character pair in the list of highest offenders. Their misrecognitions are simply all over the place.

In	Out	Freq
F	S	0.26
V	Z	0.18
8	H	0.16
P	T	0.15
A	8	0.13
Q	U	0.12
Z	V	0.10
B	E	0.09
S	F	0.09
V	B	0.08
B	D	0.08
D	B	0.07
5	I	0.07
5	Y	0.07
M	N	0.06

	Right	Wrong
A	0.82	0.18
B	0.74	0.26
C	0.91	0.09
D	0.81	0.19
E	0.92	0.08
F	0.69	0.31
G	0.93	0.07
H	0.98	0.02
I	0.96	0.04
J	0.96	0.04
K	0.94	0.06
L	0.97	0.03
M	0.92	0.08
N	0.94	0.06
O	0.97	0.03
P	0.74	0.26
Q	0.82	0.18
R	0.98	0.02
S	0.89	0.11
T	0.82	0.18
U	0.98	0.02
V	0.59	0.41
W	0.97	0.03
X	0.91	0.09
Y	0.98	0.02
Z	0.76	0.24
0	0.97	0.03
1	0.92	0.08
2	0.92	0.08
3	0.98	0.02
4	0.99	0.01
5	0.91	0.09
6	0.97	0.03
7	0.98	0.02
8	0.75	0.25
9	0.98	0.02

	Right	Wrong
V	0.59	0.41
F	0.69	0.31
P	0.74	0.26
B	0.74	0.26
8	0.75	0.25
Z	0.76	0.24
D	0.81	0.19
Q	0.82	0.18
T	0.82	0.18
A	0.82	0.18
S	0.89	0.11
5	0.91	0.09
C	0.91	0.09
X	0.91	0.09
1	0.92	0.08
M	0.92	0.08
2	0.92	0.08
E	0.92	0.08
G	0.93	0.07
K	0.94	0.06
N	0.94	0.06
J	0.96	0.04
I	0.96	0.04
0	0.97	0.03
6	0.97	0.03
O	0.97	0.03
L	0.97	0.03
W	0.97	0.03
Y	0.98	0.02
U	0.98	0.02
H	0.98	0.02
7	0.98	0.02
R	0.98	0.02
3	0.98	0.02
9	0.98	0.02
4	0.99	0.01

The first conclusion to be drawn from all this is that if at all possible, avoid the highest offender characters completely. Unfortunately, the people trying to make systems recognize alphanumeric strings rarely have influence over the people generating those strings. If there is any degree of influence, pick your battles as to how far down the list to recommend avoiding.

Nbest List Results

The next question is how useful the nbest list is in finding the right match. How far down is useful? How low should the confidence threshold be? Here are results for all 2880 transcribed utterances.

Spot	Count	Percent	Cum	Average Confidence
1	1428	49.6%	49.6%	0.76
2	418	14.5%	64.1%	0.14
3	133	4.6%	68.7%	0.03
4	83	2.9%	71.6%	0.02
5	48	1.7%	73.3%	0.01
6	24	0.8%	74.1%	0.01
7	14	0.5%	74.6%	0.01
8	13	0.5%	75.0%	0.01
9	3	0.1%	75.1%	0.01
Bad Reco	410	14.2%	89.4%	
NM	306	10.6%	100.0%	

Bad reco means an nbest list was returned, but the utterance wasn't in it. NM is a no match, meaning the recognizer couldn't even venture a guess.

In most cases, any of those average confidences for spot 2 and below would be considered too low to fool with. However, using those low confidences for nbest:2 through nbest:9 raises the total percentage of correctly found utterances from 50% to 75%. The conclusion here is to completely ignore the confidence score.

How far down the nbest list to throw against the back end is going to be highly dependent on environment. In this case, probably all the way down through 8. Check them all against the back end, then if there are multiple returns, ask the caller a second question based on other data returned to figure out which if any of them are the correct one.

In the case where the correct utterance was found somewhere in the nbest list, the average spot was 1.67 with a confidence score of 0.54.

Proactive Substitution

Returning to those dark blue cells of the highest offender misrecognitions, in the cases where the recognizer returned an nbest list but the correct utterance wasn't on it, what happens if we take nbest:1

and proactively substitute for those common misrecognitions? How often would that give us the correct utterance? There were 1158 times that this occurred where the recognizer didn't have the right utterance in the nbest list. Nbest:1 for each of these was run against two sets of proactive substitutions: the top 15 (all the dark blue) and just the top 8.

Proactive List 1
FS
VZ
8H
PT
A8
QU
ZV
BE
SF
VB
BD
DB
5I
5Y
MN

Proactive List 2
FS
VZ
8H
PT
A8
QU
ZV
BE

Take this example.

DTCUBV	BTCUDZ	D	B	T	T	C	C	U	U	B	D	V	Z

DB	**	**	**	BD	VZ

Utterances are in white, returns shaded. Three of the six characters were returned incorrectly in nbest:1. These result in the character pairs DB, BD, and VZ. All three of which are in the list of highest offenders. Where the B is returned, it is often done incorrectly for a V or a D. Proactively look both up as well as the B. Understandably, the larger the list of substitutions and the longer the string, the more possibilities this results in. For the original return of BTCUDZ, the following are the letters to check. The shaded ones get checked in the first list only.

	B	T	C	U	D	Z	
	V	P		Q	B	V	
	D						
List 1	3	2	1	2	2	2	48
List 2	1	2	1	2	1	2	8

With the first list, replacing every high offender with its common substitutions means there are now 48 strings to check against the back end. With the shorter list, there are only 8. The actual utterance correctly occurs in the list of 48, but not in the list of 8.

Run against all 1158 misrecognitions, the longer proactive substitution list results in recovering 498 or 43% of them. The shorter list results in recovering 346 or 30%.

However, the example given should highlight the downside: depending on the string, there can be many, many new strings to check. For list 1, the average was 9.3 strings to check and the maximum was 144. For list 2, the average was 4.6 and the maximum was 32. Obviously, algorithms need to be employed to limit the number of strings generated this way. Which means the full recovery probably won't be reached.

It was previously noted that when the correct string wasn't in the nbest list, nbest:2 was also recorded. This was so that proactive substitution on both nbest:1 and nbest:2 could be investigated. Running it against the nbest:2 resulted in very few additional recoveries while leading to much longer lists of strings to check.

Conclusions

So what does it all mean? What should one do with alphanumeric strings? It is now time to employ the classic voice interaction design answer of "it depends." It depends on the length of the strings. It depends on the back end: how many queries can it support at once and how long does it take to return the data. It depends on how often the strings that are tried against the back end result in a match, meaning how likely is follow-up disambiguation needed. It depends on the effectiveness of the follow-up question to correctly identify the correct string if multiple are found on the back end.

In amongst all these "it depends" are some concrete findings:

- Ignore the confidence scores. Very low ones are often right.
- Use the nbest list. Hit the back end with a bunch of possibilities.
- Do some proactive substitution above and beyond what the nbest list provides.

For Travelocity's specific problem, all of the above will be employed. The exact algorithm to be used will probably need to be tweaked, but here's the starting point. The interface with the back end supports 15 queries at once. The nbest list down to 8 will be used. Proactive substitutions in order of likelihood will be used up until the point where 30 total strings have been generated. That will result in two successive hits to the back end of 15 strings each. A follow up question or questions can then be asked to figure out which reservation is the correct one if multiples are found.