English N-Gram Language Models for Robotic Spatial Commands

Andrea Azzarone
Kungliga Tekniska Högskolan
azzarone@kth.se

Abstract

We built and testes several n-gram language models for robotic spatial commands with the help of various toolkits and frameworks such as SRILM, Julius and Wavefsurfer. We show that all these models outperform the DFA language model shipped by default in the Wavesurfer ASR Plugin. We also notice that in this particular task, with a relatively small transcriptions corpora, a simpler language model (2-gram) shows better results compared to more complex language models.

1 Introduction

The idea of human machine interaction led to research in Speech Recognition. Automatic speech recognition (**ASR**) uses the process and related technology for converting speech signals into a sequence of words or other linguistic units by means of an algorithm implemented as a computer program [1]. Recent advances in speech recognition technologies and computer hardware have made it possible to build human computer spoken dialogue systems for a wide variety of application. However, the performance of speech recognition is still a bottleneck of these systems [2]. A lot of research effort has been devoted to detecting and recovering from recognition errors.

In this work, we have tried to improve the recognition performance of the WaveSurfer ASR plugin for robotic spatial commands, by incorporating language models built on purpose. WaveSurfer ASR is a plugin that adds automatic speech recognition functionality to the WaveSurfer sound manipulation and visualisation program. The plugin is distributed as free software and is based on free resources, namely the Julius speech recognition engine and a number of freely available ASR resources for different languages [3].

Language models help a speech recognizer figure out how likely a word sequence is, independent of the acoustics. This lets the recognizer make the right guess when two different sentences sound the same. The simplest form of language model makes the strong independence assumption that words are generated independently from a multinomial distribution of dimension V, the size of the vocabulary. Such a model is called a *uni-gram language model*. *N-grams languages models* condition instead on the previous *n-1* terms. In addition to n-gram language models, Julius supports Deterministic Finite Automaton (DFA) grammars/language models. A grammar of this type is the one used by default by the ASR Wavesurfer Plugin. The default grammar actually corresponds to a uni-gram language model where each word has the same probability to be hit.

We will consider the problem of constructing n-gram language models from a set of 5863 robotic spatial commands built upon the SemEval-2014 Task 6 data sets[1]. Some examples or robotic spatial commands are shown in table 1.

[1] http://alt.qcri.org/semeval2014/task6/

Table 1: Examples from our corpora

1	Pick up the red cube.
2	Drop the red prism.
3	Move the blue cube to the top of the single green cube.
4	Put yellow pyramid on yellow block.

2 Method

2.1 Preprocessing

The command sentences are pre-processed to remove punctuation and to put all words in upper-case in order to match with the case of the dictionary provided in the ASR Wavesurfer Plugin (*full.dict*).

2.2 N-Gram Languages Models

In an n-gram model, the probability $P(w_1, \ldots, w_m)$ of observing the sentence $w_1, \ldots, w_m$ is approximated as

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) \tag{1}$$

Here, it is assumed that the probability of observing the i^{th} word w_i in the context history of the preceding $i-1$ words can be approximated by the probability of observing it in the shortened context history of the preceding $n-1$ words (n^{th} order Markov property).

The conditional probability can be calculated from n-gram model frequency counts:

$$P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1})} \tag{2}$$

Typically, however, the n-gram model probabilities are not derived directly from the frequency counts, because models derived in this way have severe problems when confronted with any n-grams that have not explicitly been seen before. Instead, some form of smoothing is necessary, assigning some of the total probability mass to unseen words or n-grams. Various methods are used, from simple "add-one" smoothing (assign a count of 1 to unseen n-grams) to more sophisticated models, such as Witten-Bell discounting or Chen and Goodman's modified Kneser-Ney discounting. In general, it is also useful to interpolate higher-order n-gram models with lower-order n-gram models, because when there is insufficient data to estimate a probability in the higher-order model, the lower-order model can often provide useful information [4].

To build the language models we use The SRI Language Modeling Toolkit (SRILM) [5]. We builded several language models with different orders (from 2 to 6), discounting techniques, with or without interpolation. A list of language models built is shown in table 2. We used the **ngram-count** command of SRILM toolkit to automatically create a language model from a training corpora.

We further processed the language models to produce binary language models compatible with Julius. We used the utility **mkbingram** from the Julius toolkit [6].

3 Experiments

We recorded 15 robot command utterances sampled at $16KHz$ using the **arecord** utility in a noise environment. Each utterance is recorded 3 times for a total of 45 testing utterances. The speaker is a non-native English male speaker. The transcriptions of the utterances can be find in table 3. The utterances can contain words not originally present in the commands used to train the language models.

Using the **julius** command from the Julius toolkit we got the speech-to-text transcriptions for all the utterances using all the languages model built previously. We also compared our languages model

Table 2: Languages models built using SRILM

Order	Discounting		Interpolation
	Witten-Bell	Chen and Goodman	
2-grams			
2-grams	✓		
2-grams	✓		✓
2-grams		✓	
2-grams		✓	✓
⋮	⋮	⋮	⋮
6-grams			
6-grams	✓		
6-grams	✓		✓
6-grams		✓	
6-grams		✓	✓

with the default English *DAF* grammar provided with the *ASR Wavesurfer plugin*. We post-processed the transcriptions in order to remove the <*s*> and </*s*> markups.

We used the average Word Error Rate (**WER**) as a measure of the accuracy. WER is a common metric of the performance of a speech recognition or machine translation system and can be easily calculated as:

$$WER = \frac{S + D + I}{N} \tag{3}$$

where *S* is the number of substitutions, *D* is the number of deletions, *I* is the number of insertions and *N* is the number of words in the reference.

Table 3: Transcription of testing utterances

1	Hold the box
2	Put the red cube
3	Place the blue brick on top of the red brick
4	Place the yellow brick on the green cube
5	Pick up cube
6	Pick up the red box
7	Move one yellow block onto the green block
8	Pick the left blue block
9	Drop the cube
10	Move the yellow block on top of the red block
11	Drop the box
12	Move the table
13	Drop the block in front of the cube
14	Pick the cube
15	Place the blue cube in front of the yellow cube

4 Results

In the speech recognition phase *julius* is used to recognize all the recorded 45 utterances with each previously built language model. We also tested the default DFA grammar in order to understand if our language models improve the accuracy of the ASR system. A summary of the obtained WERs is shown in table 4.

Table 4: WER for each language model

Order	Discounting				
	None	Witten-Bell		Chen and Goodman	
		w/o interpolation	w/interpolation	w/o interpolation	w/interpolation
2-grams	**39.03**%	59.16%	58.33%	73.67%	75.34%
3-grams	44.40%	59.24%	59.72%	74.14%	71.35%
4-grams	39.24%	55.90%	60.13%	68.25%	70.73%
5-grams	44.05%	57.90%	60.13%	68.63%	71.52%
6-grams	44.05%	57.90%	60.13%	68.63%	71.52%
dfa	91.41%				

5 Discussion and Conclusions

We built and tested several n-gram language models for robotic spatial commands with the help of various toolkits and frameworks (such as SRILM, Julius and Wavefsurfer) starting from a set of 5863 robotic spatial commands built upon the SemEval-2014 Task 6 data sets. We experimented multiple orders (from 2-grams languages models up to 6-grams), two different discounting techniques, with and without interpolation. 45 test utterances have been recorded in a noise environment from a non-native English speaker (15 commands with 3 repetitions for each one). As we can see in table 4, the 2-gram model (without discounting nor interpolation) seems to outperform all the other models. This is likely due to the relatively small dimension of our training corpora and to its simplicity. Smoothing and interpolation would probably help in case our test utterances would contain more words not belonging to our training corpora. As expected all the built languages model outperform the DFA grammar shipped by default in the ASR Wavesurfer Plugin.

The relative high WER for all the models is likely due to the fact that the utterances have been recorded from a non-native English speaker in a noise environment and the acoustic models shipped by default in the ASR Wavesurfer plugin are probably not properly trained for this kind of scenario. It would be helpful for future works to use better acoustic models trained with utterances from speakers of different countries. A bigger corpora of spatial robotic commands it's also required. It would be interesting to write a probabilistic context-free grammar (**PCFG**) in order to generate these commands and to test n-gram languages models against a DFA language model built starting from the PCFG. Reducing the size of the vocabulary should also improve the accuracy of all the models. We keep all this idea for further research.

References

[1] S. Swamy and K. Ramakrishnan, "An efficient speech recognition system," *Computer Science and Engineering: An International Journal (CSEIJ)*, vol. 3, August 2013.

[2] D. J. Litman and S. Pan, "Empirically evaluating an adaptable spoken dialogue system," *CoRR*, vol. cs.CL/9903008, 1999.

[3] G. Salvi and N. Vanhainen, "The wavesurfer automatic speech recognition plugin," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)* (N. C. C. Chair), K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, eds.), (Reykjavik, Iceland), European Language Resources Association (ELRA), may 2014.

[4] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, (Stroudsburg, PA, USA), pp. 310–318, Association for Computational Linguistics, 1996.

[5] A. Stolcke, "Srilm-an extensible language modeling toolkit," in *Proceedings International Conference on Spoken Language Processing*, pp. 257–286, November 2002.

[6] A. Lee and T. Kawahara, "Recent development of open-source speech recognition engine julius," 2009.