Lightly Supervised Learning of Text Normalization: Russian Number Names

Abstract

Most areas of natural language processing today make heavy use of automatic inference from large corpora. One exception is text-normalization for such applications as text-to-speech synthesis, where it is still the norm to build grammars by hand for such tasks as handling abbreviations or the expansion of digit sequences into number names. One reason for this, apart from the general lack of interest in text normalization, has been the lack of annotated data. For many languages, however, there is abundant *unannotated* data that can be brought to bear on these problems. This paper reports on the inference of number-name expansion in Russian, a particularly difficult language due to its complex inflectional system. A database of several million spelled-out number names was collected from the web and mapped to digit strings using an overgenerating number-name grammar. The same overgenerating number-name grammar can be used to produce candidate expansions into number names, which are then scored using a language model trained on the web data. Our results suggest that it is possible to infer expansion modules for very complex number name systems, from unannotated data, and using a minimum of hand-compiled seed data.

1 Introduction

This paper deals with an *understudied area* in NLP, namely text normalization, in particular for such applications as text-to-speech synthesis or automatic speech recognition. Partly because the area is understudied, it is one of the few areas where complex hand-built systems still play a major role. While some work has been done on applying data driven approaches to such problems as abbreviation expansion (Sproat et al., 2001; Olinsky and Black, 2000), many problems, including seemingly mundane things like the expansion of digit-strings into number names, still rely on hand-engineered grammars.

For number-name expansion in particular, part of the reason for this is that on the one hand, it is relatively easy (for some languages) to write a piece of code (or build a finite-state transducer) that maps from sequences such as *2,423* into number names such as *two thousand four hundred and twenty three*; whereas on the other hand it is hard to find enough training data that pairs digit strings with their number-name expansion. But how about inducing number name expansion from *unannotated* text?

Here we report on some experiments on addressing this problem for Russian, a language with a particularly complex number-name system. Designing a model to expand a digit string into a well-formed number name in Russian is significantly more complicated than the comparable problem for English, due to the complex case and gender system of Russian. Furthermore, choosing the appropriate expansion depends upon context.

In this paper we will present a simple procedure for mining Russian number names from the Web, and learning the mapping between digit strings and number names. We will compare both generative (n-gram) language models and two discriminative methods and show, that *at least for the methods tried*, the n-gram language model yields the best performance.

Furthermore, we will show that nearly all of the errors in the learned system relate to problems with selecting the right contextual expansion, which is expected to be difficult; and that very few errors are due to number names that are *internally* ill-formed. Since it is precisely the internal well-formedness that is easiest to capture with a set of hand-constructed rules, this result suggests that such rules may be dispensible, and that the internal well-formedness of a number name should be handled by a language model that also selects the contextually most appropriate form.

2 Russian Numbers

Russian (along with other highly inflected Slavic languages such as Czech, or Croatian) has what is probably the most complex number name system of any language. The complexity is due to the case, number and gender marking on nouns and adjectives, which also carries over to the number system. Russian distinguishes two numbers (singular, plural), three genders (masculine, feminine, neuter) and six cases (nominative, accusative, genitive, dative, prepositional and instrumental). In general, numbers agree in gender with the nouns they modify. Thus один город *one city* has *one* in the masculine nominative/accusative, but одна собака *one dog*, has *one* in the feminine, following the gender of the nouns. Similarly for два города *two cities*, versus две собаки *two dogs*. In an oblique case, such as the instrumental, the numeral must agree with the noun in case: в двух шагах *at two paces*. Complex numerals decline in their entirety: к тремстам тридцати шести часам *to three hundred and thirty six hours* (dative case); с пятью тысячами пятьюстами семьюдесятью четырьмя рублями *with five thousand five hundred and seventy four rubles* (instrumental case). This short description only begins to scratch the surface of the complexity: the reader is referred to any good pedagogical grammar of Russian, such as (Wade, 1992) for details.

To get a sense of the range of forms one finds for a single number in written text, consider Table 1, which are various contextually appropriate renditions of *two*, *three thousand* and *twenty*, culled from Russian web pages.

3 Methodology

The basic method outlined here can be summarized in the following steps:

- Provide a seed-list **L** of all legal forms of single-word number terms.
- Mine web pages for sequences of terms from **L**, along with their contexts.
- Using a loose number-name grammar, filter the resulting list for combinations that fit the general properties expected of well-formed number names; the grammar is implemented as a finite-state transducer, which will accept only reasonable-looking number names, and map them to their corresponding digit sequences.
- The result of the previous step is a large list of annotated digit-string/number-name pairs in context. We now use these data to train a model that will produce a contextually appropriate number name expansion given a digit string.

In what follows we detail each of these steps.

3.1 Providing the seed list

The first step is to provide a seed list of basic number terms in all of their inflected forms. Number forms for the words for *1*, *10*, *50*, *100* and *500* for Russian would look as in Table 2. In the case of *1*, the forms reflect differences in gender, case and number; for the other numbers listed here, the only marked differences are for case. For, example, пятьдесять is the Nominative/Accusative case form, пятидесяти is the Genitive/Dative/Prepositional form, and пятьюдесятью is the Instrumental form.

Ideally such lists of forms could be mined from web pages that deal with Russian grammar, but this is likely to be difficult to achieve because grammatical descriptions of languages as complex as Russian rarely simply list all the forms, and instead depend upon the reader to construct the complete set from information given elsewhere in the grammar. A fairly typical example from a good pedagogical grammar of Russian is the following. So, Wade (1992, p. 197) notes that “тысяча ‘thousand’ declines like second-declension дача ‘country cottage’

Left Context	*Number*	*Right Context*
наступающие через день	два	в случае когда
явное предпочтение отдаётся	двум	последним это единственные
бухаре кроме того	две	новые гостиницы планируется
уровень на базе	двух	резервируемых станций арм
и получал около	трех тысяч	рублей в месяц
уже как минимум	три тысячи	лет поэтому установить
асбест применяется в	трех тысячах	наименований материалов и
авиабилеты потребовались сразу	трем тысячам	людей участники несостоявшегося
поп культуры в	двадцать пять	он начал печататься
прибли зительно о	двадцати пяти	пророках и посланниках
ограничить портфель госхолдинга	двадцатью пятью	крупными компаниями что

Table 1: Contextually appropriate renditions of *2*, *3,000* and *20*, from Russian web pages. (Unfortunately lack of space does not allow for glosses.)

", so in order to find all the forms of тысяча one would have to consult a table that lists all the forms of second-declension nouns like дача. Even if resources as good as a traditional grammar such as Wade's could be found on the web, it is doubtful that one could perform the correct inference to produce all and only the correct forms of тысяча. Thus, instead, a list consisting of 187 entries for forms of single-word number names ranging in value between 1 and 10^{12} was constructed by hand.

1	один	10	десять
1	одного	10	десяти
1	одному	10	десятью
1	одним	50	пятьдесять
1	одном	50	пятидесяти
1	одна	50	пятьюдесятью
1	одну	100	сот
1	одной	100	ста
1	одною	500	пятьсот
1	одно	500	пятисот
1	одни	500	пятистам
1	одних	500	пятьюстами
1	одними	500	пятистах

Table 2: Sample seed items.

3.2 Mining web pages

We collected all instances of sequences of one or more words from the number name list on all Russian-language pages from a snapshot of the Web. Also collected was a window of three tokens on either side of the number name.

3.3 The number-name grammar and filtering

While languages show a lot of variation in their expression of numbers, there are certainly constraints on what is not possible, or at least very unlikely. The most extensive study of linguistic constraints on number-name formation is (Hurford, 1975) (but see also (Brandt Corstius, 1968) for earlier work.) For example, in decimal number-name systems, it is normal for small major powers of ten to precede large major powers when these are in a multiplicative relation. Thus in English we have *two hundred million*, but not *two million hundred*, though these would in principle evaluate to the same integer.

Based on previous work such as (Hurford, 1975) one can write a covering grammar that allows reasonable combinations of basic number name terms, but disallows sequences that are probably ill-formed. The grammar should have the property that it would admit *two hundred million* as an expression of $2{\times}10^8$, but if the word sequence *two million hundred* were found that would not be treated as a number name. The grammar is based on the approach to number names described in (Sproat, 1997). In that approach, number names were modeled as the composition of two basic FSTs. The first is a *factorization* FST $\mathcal{F}$, which maps from digit sequences (up to some bounded length) to sums of products of powers of ten; the second is a *lexicon* FST $\mathcal{L}$, which maps from sums of products of powers of ten to number words. The map between a digit sequence and a number name can then be modeled as $\mathcal{F} \circ \mathcal{L}^*$. Thus given the following fragment of an English lexicon

2	two
$2{\times}10^1$	twenty
10^2	hundred

the factorization of *222* as $2{\times}10^2$ $2{\times}10^1$ 2 combined with the lexicon would yield *two hundred twenty two*. In practice we generally

два десять	двум десяти
два десяти	двум десятью
два десятью	двумя десять
две десять	двумя десяти
две десяти	двумя десятью
две десятью	двум десять
двух десять	<u>двадцать</u>
двух десяти	<u>двадцати</u>
двух десятью	<u>двадцатью</u>

Table 3: Initial renditions of *20*.

also need a few cleanup rules, implemented as additional transducers, to handle things like use of *and*, or respelling of numbers in certain combinations. For larger numbers, the factorization will depend upon which powers of the base exist as basic terms in the target language. In most Western languages there is no word for 10^4: we say *ten thousand*. So we want to factor *10,000* as $1{\times}10^1{\times}10^3$. Chinese, along with other East Asian languages, on the other hand does have a word for 10^4, and so for Chinese we want to factor *10,000* as $1{\times}10^4$. Chinese lacks a single word for 10^5, so that *100,000* would be represented as $1{\times}10^1{\times}10^4$. South Asian (Indian) languages, however, *do* have a word (*lakh*), so that for those languages *100,000* would be $1{\times}10^5$.

For the present experiment, we assume we do not know which type of grammar Russian has, and so we use the lexicon that we created at the outset in combination with the *union* of the FSTs representing Western, East Asian, and South Asian factorization grammars. Our grammar also makes no assumptions about *blocking* as described above, so that we do not assume that *two ten* will be blocked by the existence of *twenty*. This grammar naturally overgenerates, but the grammar will be constrained on the number-name side by the word sequences that are actually observed on the web. For *20*, the grammar will allow any of the examples in Table 3. Of these only the last three underlined renditions are correct, the others being illicit combinations of *two* plus *ten*, which either do not occur or are in any case rare.

3.4 The resulting dataset

The result of filtering the web data with the grammar described in the previous section is a list of nearly 26 million number names in context.

3.5 Training and testing models.

We evaluated both generative (n-gram) language models as well as discriminative models — perceptron and decision lists — on the problem of constructing well-formed number names in context. We describe each of these in turn.

3.5.1 N-gram language model

We selected from our web data 7.5 million examples of Russian number names in context, comprising 60 million words. From these data we constructed a trigram language model using Kneser-Ney smoothing.

There were two sets of test data. One, which we term *token balanced*, the other *type balanced*. The *token balanced* set consists of 1,000 examples randomly selected from a held-out portion of the web corpus. The distribution of number names is therefore fairly representative with the distribution of number names one finds in Russian text on the web. (See Section 5.) For the *type balanced* set we select exactly one instance of each number name type so that we have a good sample of the different types of number names found, irrespective of their frequency: this resulted in a test set with 826 examples.

In both cases, the test set was processed by replacing the number names with their digit representation, and then using the number-name expansion FST described earlier to map back to all possible expansions of the number name in question. As we saw with the example of *20* above, this will result in legal (though not necessarily contextually-appropriate) expansions, as well as illegal expansions. Thus for each test phrase, we have a lattice of possible expansions of the number in that phrase. The language model is then intersected with the lattice, and the lowest cost path selected. The number of pre-intersection candidates in the lattice of course varies greatly depending upon the length of the number. For *1*, 14 candidates are produced; for *235,038*, for instance, 405,000 candidates are produced. The technique we have just described is similar in spirit to work on generation that uses statistical language modeling — e.g. (Langkilde and Knight, 1998), though the problem domain presented here is, as far as we know, novel.

3.5.2 Perceptrons and Decision Lists

In order to compare the n-gram language model's performance with other possible methods, we selected a subset of the test data that had number names consisting of single words. Here the problem is merely to select the contextually appropriate word, without the additional problem of modeling well-formedness within the number name. The reason for picking single-word number names was that it is easier to model this as a classification problem: most single-word numbers occur frequently enough that one can expect to have seen most or all possible forms of the word. In any event, if the discriminative methods work well on the single-word number names, there will be a case for extending the methods to handle all number names, including developing methods for determining the correct internal form of long complex number-names. There were a total of 7.36 million training examples, though there were a wide range of counts from as low as 23 examples (for *80*) to as high as 3.3 million examples (for *1*).

We used two techniques. The first was a perceptron, trained using the SNoW machine learning package (Carlson et al., 1999), which provides for multi-class classification. For the perceptron learning we set α (the learning rate) to 0.1, the threshold θ was set to 4.0 and the initial feature weight was 0.2. α affects the speed of update of features active in positive ($\mathcal{P}_t$) and negative ($\mathcal{N}_t$) examples:

$$\forall i \in \mathcal{P}_t, w_{t,i} \leftarrow w_{t,i} + \alpha_t s_i$$
$$\forall i \in \mathcal{N}_t, w_{t,i} \leftarrow w_{t,i} - \alpha_t s_i$$

The threshold, θ, is a parameter of the sigmoid function:

$$\sigma(\theta, \Omega) = \frac{1}{1+e^{\theta-\Omega}}$$

For each number name, the contextual features were as follows:

- The word to the immediate left/right of the number (L1, R1)
- Each other word in the left/right context, tagged as being in the left/right context
- The bigram to the immediate left, spanning, and right of the number
- The two-character suffix of the word to the left, and the word to the right of the number; these features are intended as crude morphological information

Naturally one could conceive of many other features that could be added, but note that with the lexical features, this already constitutes hundreds or thousands of features for any particular training set, and that this is a *superset* of the features available to the trigram language model: the bigram features listed above are exactly the features that the trigram language model would use to disambiguate these cases. Also, note that *the vast majority* of the 7.5 million number names used to train the trigram LM are single word number names: 7.36 million, as we noted above. The trigram LM could only have had a slight advantage from seeing extra data.

We built disambiguation models for each of the single-word number names (Table 6). The training data were all examples for each of these numbers found in the 7.5 million number name set described above, with the possible expansions of each of these numbers being the classes to be predicted by the perceptron.

In addition, we also trained a decision list (Yarowsky, 1996), using the same features as for the perceptron. For each feature j, we compute $\mathrm{P}(\mathrm{C}_i|\mathrm{F}_j)$, the probability of the ith class given that feature. Assuming class k is the most probable class with F_j, we compute

$$Abs[\frac{Log(P(C_k|F_j))}{P(C_{\overline{k}}|F_j)}]$$

(where $\overline{k}$ is the complement of k) and then sort the entire list of features by this log likelihood value. The features at the beginning of the list are the ones that are most discriminative. Following Yarowsky, in test mode the class prediction for the *first* feature that matches is the one that is picked.

The test data for these experiments were as many as 1,000 instances for each number, though several had fewer instances (see Table 6). For comparison, the trigram language model was also tested on these examples.

4 Results

4.1 Trigram language model

Tables 4 and 5 present the results for the type-balanced and token-balanced tests using the trigram language model. Each table lists the

Total forms:	826
Total words:	1937
Total multiword numbernames:	680
Total long multiword numbernames:	49
Form error rate:	0.23
Word error rate:	0.14
Form accuracy:	0.77
Imposs. form err.:	0.01
Imposs. form err., multiword:	0.02
Imposs. form err., long multiword:	0.08

Table 4: Type balanced test

Total forms:	1000
Total words:	1021
Total multiword numbernames:	16
Total long multiword numbernames:	1
Form error rate:	0.12
Word error rate:	0.12
Form accuracy:	0.88
Imposs. form err.:	0.00
Imposs. form err., multiword:	0.00
Imposs. form err., long multiword:	0.00

Table 5: Token balanced test

total forms (in context) tested, the total number of words in the tested number names, the number of multiword number names, the number of long multiword number names (five or more words long), the form error rate (i.e., a number name is wrong if *any* word in it is wrong), the *word* error rate and the form accuracy (one minus the error rate). The final three rows represent errors that involve expansions that are impossible in any context — in other words, number names that are internally ill-formed. Examples would include case or gender mismatches within the number name. These are broken down into all cases of such errors, cases that involve multi-word number names, and cases that involve long multiword numbernames. Note that the impossible form rates are actually an upper bound: a form was determined to be impossible if this particular expansion for the number in question was never found among the data collected from the web: this does not rule out the possibility that the combination is in fact legal, just never observed. Clearly error rates are higher in the type-balanced than the token-balanced test, as one would expect: the type-balanced text contains a higher proportion of rarer and more complex number names than the token balanced test. Furthermore the error rates are still fairly high, even in the token balanced test, which has an overall error rate of 0.12. But the point to notice is that the impossible form rates are quite low, only as high as 0.08 in the type-balanced test. This is significant, since what it means is that nearly all the errors relate to the problem of picking the appropriate number name expansion for the context, rather than the internal well-formedness of complex number names. And this suggests in turn that even a perfect grammar of number names, would only improve performance slightly: most of the problem of Russian number names is determining which form to use in the given context.

4.2 Comparison of all methods

The performance of the perceptron, decision list, and the trigram language model on the single-word number names is shown in Table 6. The lefthand column gives the digit sequence expanded into the number name, and the remaining columns list accuracy for each of the methods. Interestingly, the trigram language model generally outperforms either of the discriminative methods, which is perhaps surprising since the discriminative methods in principle had access to a wider selection of features than the trigram language model. Some of the cases where at least one of the discriminative methods seemed to outperform the trigram language model were in the teens, but here the difference is an artifact. Recall that the number grammar allowed illegal expansions such as десять четыре *ten four* for *14* (the correct form being четырнадцать). The trigram language model was required to consider these illegal expansions too, and occasionally scored these higher than the correct expansion. In contrast, the discriminative methods, since they were trained only on data that involved *single-word* expansions, never had an option to expand *14* in this way, and so were saved from making this error. Again, it is perhaps surprising that the discriminative methods did not in general outperform the trigram language model. It is of course possible that with different parameter settings or a different set of features than the ones provided the perceptron, for instance, could perform better than the results shown here. But as they stand the results do not suggest that there is likely to be a big gain from using these discriminative

Number	# Test examples	Perceptron	Decision List	Trigram LM
1	1000	0.79	0.68	**0.83**
2	1000	0.94	0.86	**0.95**
3	1000	0.96	0.90	**0.97**
4	1000	0.95	0.90	**0.98**
5	1000	0.92	0.89	**0.96**
6	1000	0.90	0.85	**0.94**
7	1000	0.87	0.80	0.**92**
8	1000	0.90	0.85	**0.92**
9	1000	0.90	0.87	**0.94**
10	1000	0.92	0.88	**0.95**
11	289	0.85	0.81	**0.93**
12	428	**1.00**	0.97	0.99
13	165	**0.97**	0.96	0.96
14	119	**0.99**	0.98	0.97
15	402	**1.00**	0.99	0.98
16	107	**0.97**	0.95	0.96
17	115	**0.98**	0.97	0.97
18	96	**1.00**	0.98	0.97
19	43	1.00	1.00	0.93
20	1000	0.93	0.91	**0.95**
30	746	0.90	0.87	**0.94**
40	796	0.90	0.85	**0.92**
50	184	0.97	0.97	0.94
60	67	1.00	1.00	0.99
70	57	0.98	0.98	0.98
80	1	1.00	1.00	1.00
90	114	0.89	0.87	0.89
100	1000	0.94	0.90	**0.96**
200	353	0.90	0.87	**0.92**
300	271	0.91	0.86	**0.93**
400	68	0.96	0.91	**0.97**
500	164	0.80	0.76	**0.91**
600	56	0.94	0.89	**0.98**
700	30	**1.00**	0.97	0.90
800	25	**0.92**	0.80	0.88
900	24	0.96	0.96	0.96
1000	1000	0.66	0.76	**0.79**
1000000	1000	0.66	0.63	**0.71**
1000000000	1000	0.59	0.62	**0.67**
1000000000000	10	**0.60**	0.50	0.50

Table 6: Comparison of methods. If a single method is a clear winner, this is marked in bold.

models for this particular task.

5 Coverage issues

The data we have worked with consists of number names that are written out as words, where the task was to reconstruct those words from a digit representation of the same numbers. But do people tend to write the same kinds of numbers as words as they write with digits? Is the sample we have considered a representative sample of what *actually* would need to be expanded into words, and therefore is the performance we see here likely to be representative? The answer seems to be, reasonably so, with some qualifications. Figure 1, left panel, shows the distribution of numbers in our sample. On the horizontal axis is the numerical value of the number name, on the vertical axis its count, plotted on a log-log scale. Some patterns immediately become apparent. The highest counts in any region are for the powers of ten that are represented as a single word in Russian, in particular 10^2, 10^3, 10^6, 10^9, 10^{12}. A recurring pattern with these powers is that the base power (1×10^6, for example) is roughly two orders of magnitude more frequent than the next multiplier (e.g. 2×10^6). 10^2 is different in this regard, but then 2×10^2 and so forth are written as single words in Russian. Turning to the right panel in Figure 1, we see the distribution of the same number values, if they occur written as digits: note that such data are inherently noisy since especially for longer strings of digits, we cannot be sure that the digit was intended to be read as a number name (as opposed to as a string of individual

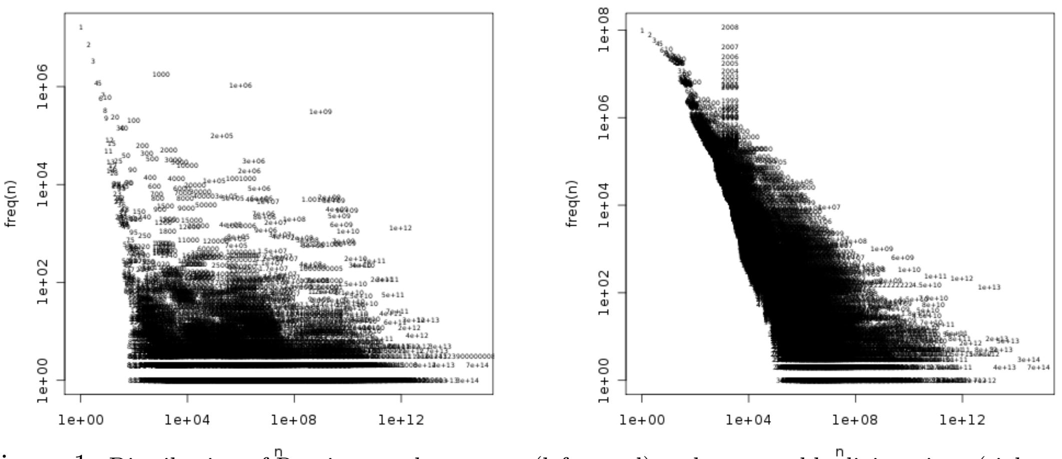

Figure 1: Distribution of Russian number names (left panel) and comparable digit strings (right panel).

numbers, as in the normal reading of telephone numbers in American English). Nevertheless, the distribution is broadly similar to the distribution in Figure 1. The main differences are as follows. First, the plot is less scattered. In numbers written as words, there is likely a resistance against writing long numbers that would result in lots of words, so that something like *1,234,322* is relatively unlikely to be written out, compared to *1,000,000*, which can be written as a single word. If the number is written as a digit, it makes no difference in terms of length which digits are used. Presumably as a result of this, numbers like 2×10^6 are less frequent than 1×10^6 by a smaller amount. Second, there is a disproportionate number of instances of numbers around *2,000*, which of course are years.

So these data suggest that while there are differences in the distributions of numbers written as words versus as digit sequences, the distributions are not wildly different. Thus the results reported here are reasonably representative of what we would find in applying the models to cases originally written with digits.

6 Future work

Number names are complicated, and in no language is this more true than in Russian. We have shown that using a few linguistically motivated constraints, combined with standard language modeling techniques, we can infer a system that does quite well at the job of expanding from digit strings to number names.

There is still more work to be done even on Russian: we would like to improve performance over what we see here, so more sophisticated features – but ones that require minimal linguistic knowledge beforehand – should be investigated. Clearly other discriminative methods beyond the ones tried here should also be investigated; some plausible possibilities would be averaged perceptron (Collins, 2002) or logistic regression (Hastie et al., 2009). Furthermore, there are constructions that the current data does not cover. One case discussed in (Sproat, 1997) is the expansion of numbers before words like процент *percent*: when modifying another noun (*7 percent solution*) the word for percent in Russian must take an adjectival form, agreeing with the following noun in case, number and gender. The number that occurs before that must itself be in a genitive form.

The linguistic constraints that we have incorporated into the finite-state grammar are not as general as they would need to be to cover number name constructions in other languages. For example, the vigesimal system of traditional Welsh number names presents a real challenge. In the traditional system, *99* is expressed as *pedwar ar bymtheg a phedwar ugain*, literally *four on fifteen and four twenties*. To capture such cases — but not let in cases that do not occur in number name systems — will require further tuning of the number-name grammar.

References

Hugo Brandt Corstius, editor. 1968. *Grammars for Number Names*. Number 7 in Foundations of Language, Supplementary Series. D. Reidel, Dordrecht.

Andrew J. Carlson, Chad M. Cumby, Jeff L. Rosen, Dan Roth, and Nicholas D. Rizollo. 1999. SNoW user guide. Technical report.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *Elements of Statistical Learning*. Springer, 2nd edition.

James Hurford. 1975. *The Linguistic Theory of Numerals*. Cambridge University Press, Cambridge.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus knowledge. In *COLING/ACL*.

Craig Olinsky and Alan Black. 2000. Non-standard word and homograph resolution for Asian language text analysis. In *ICSLP-2000*, Beijing, China.

Richard Sproat, Alan Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of non-standard words. *Computer Speech and Language*, 15(3):287–333.

Richard Sproat, editor. 1997. *Multilingual Text to Speech Synthesis: The Bell Labs Approach*. Kluwer Academic Publishers, Boston, MA.

Terence Wade. 1992. *A Comprehensive Russian Grammar*. Blackwell, Oxford.

David Yarowsky. 1996. *Three Machine Learning Algorithms for Lexical Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania, Philadelphia.