**ngram-count**

**NAME**

ngram-count - count N-grams and estimate language models

**SYNOPSIS**

**ngram-count** [ **-help** ] *option* ...

**DESCRIPTION**

**ngram-count**generates and manipulates N-gram counts, and estimates N-gram language models from them. The program first builds an internal N-gram count set, either by reading counts from a file, or by scanning text input. Following that, the resulting counts can be output back to a file or used for building an N-gram language model in ARPA [ngram-format(5)](http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html). Each of these actions is triggered by corresponding options, as described below.

**OPTIONS**

Each filename argument can be an ASCII file, or a compressed file (name ending in .Z or .gz), or ``-'' to indicate stdin/stdout.

**-help**

Print option summary.

**-version**

Print version information.

**-order***n*

Set the maximal order (length) of N-grams to count. This also determines the order of the estimated LM, if any. The default order is 3.

**-vocab***file*

Read a vocabulary from file. Subsequently, out-of-vocabulary words in both counts or text are replaced with the unknown-word token. If this option is not specified all words found are implicitly added to the vocabulary.

**-vocab-aliases***file*

Reads vocabulary alias definitions from *file*, consisting of lines of the form

*alias* *word*

This causes all tokens *alias*to be mapped to *word*.

**-write-vocab***file*

Write the vocabulary built in the counting process to *file*.

**-write-vocab-index***file*

Write the internal word-to-integer mapping to *file*.

**-tagged**

Interpret text and N-grams as consisting of word/tag pairs.

**-tolower**

Map all vocabulary to lowercase.

**-memuse**

Print memory usage statistics.

**Counting Options**

**-text***textfile*

Generate N-gram counts from text file. *textfile*should contain one sentence unit per line. Begin/end sentence tokens are added if not already present. Empty lines are ignored.

**-text-has-weights**

Treat the first field in each text input line as a weight factor by which the N-gram counts for that line are to be multiplied.

**-no-sos**

Disable the automatic insertion of start-of-sentence tokens in N-gram counting.

**-no-eos**

Disable the automatic insertion of end-of-sentence tokens in N-gram counting.

**-read***countsfile*

Read N-gram counts from a file. Ascii count files contain one N-gram of words per line, followed by an integer count, all separated by whitespace. Repeated counts for the same N-gram are added. Thus several count files can be merged by using [cat(1)](http://www.speech.sri.com/projects/srilm/manpages/cat.1.html) and feeding the result to **ngram-count -read -** (but see [ngram-merge(1)](http://www.speech.sri.com/projects/srilm/manpages/ngram-merge.1.html) for merging counts that exceed available memory). Counts collected by **-text**and **-read**are additive as well. Binary count files (see below) are also recognized.

**-read-google***dir*

Read N-grams counts from an indexed directory structure rooted in **dir**, in a format developed by Google to store very large N-gram collections. The corresponding directory structure can be created using the script **make-google-ngrams**described in [training-scripts(1)](http://www.speech.sri.com/projects/srilm/manpages/training-scripts.1.html).

**-intersect***file*

Limit reading of counts via **-read**and **-read-google**to a subset of N-grams defined by another count *file*. (The counts in *file*are ignored, only the N-grams are relevant.)

**-write***file*

Write total counts to *file*.

**-write-binary***file*

Write total counts to *file*in binary format. Binary count files cannot be compressed and are typically larger than compressed ascii count files. However, they can be loaded faster, especially when the **-limit-vocab**option is used.

**-write-order***n*

Order of counts to write. The default is 0, which stands for N-grams of all lengths.

**-write***n file*

where *n*is 1, 2, 3, 4, 5, 6, 7, 8, or 9. Writes only counts of the indicated order to *file*. This is convenient to generate counts of different orders separately in a single pass.

**-sort**

Output counts in lexicographic order, as required for [ngram-merge(1)](http://www.speech.sri.com/projects/srilm/manpages/ngram-merge.1.html).

**-recompute**

Regenerate lower-order counts by summing the highest-order counts for each N-gram prefix.

**-limit-vocab**

Discard N-gram counts on reading that do not pertain to the words specified in the vocabulary. The default is that words used in the count files are automatically added to the vocabulary.

**LM Options**

**-lm***lmfile*

Estimate a language model from the total counts and write it to *lmfile*. This option applies to several LM model types (see below) but the default is to estimate a backoff N-gram model, and write it in [ngram-format(5)](http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html).

**-write-binary-lm**

Output *lmfile*in binary format. If an LM class does not provide a binary format the default (text) format will be output instead.

**-nonevents***file*

Read a list of words from *file*that are to be considered non-events, i.e., that can only occur in the context of an N-gram. Such words are given zero probability mass in model estimation.

**-float-counts**

Enable manipulation of fractional counts. Only certain discounting methods support non-integer counts.

**-skip**

Estimate a ``skip'' N-gram model, which predicts a word by an interpolation of the immediate context and the context one word prior. This also triggers N-gram counts to be generated that are one word longer than the indicated order. The following four options control the EM estimation algorithm used for skip-N-grams.

**-init-lm***lmfile*

Load an LM to initialize the parameters of the skip-N-gram.

**-skip-init***value*

The initial skip probability for all words.

**-em-iters***n*

The maximum number of EM iterations.

**-em-delta***d*

The convergence criterion for EM: if the relative change in log likelihood falls below the given value, iteration stops.

**-count-lm**

Estimate a count-based interpolated LM using Jelinek-Mercer smoothing (Chen & Goodman, 1998). Several of the options for skip-N-gram LMs (above) apply. An initial count-LM in the format described in [ngram(1)](http://www.speech.sri.com/projects/srilm/manpages/ngram.1.html) needs to be specified using **-init-lm**. The options **-em-iters**and **-em-delta**control termination of the EM algorithm. Note that the N-gram counts used to estimate the maximum-likelihood estimates come from the **-init-lm**model. The counts specified with **-read**or **-text**are used only to estimate the smoothing (interpolation weights).

**-maxent**

Estimate a maximum entropy N-gram model. The model is written to the file specifed by the **-lm**option.   
If **-init-lm***file* is also specified then use an existing maxent model from *file*as prior and adapt it using the given data.

**-maxent-alpha***A*

Use the L1 regularization constant *A*for maxent estimation. The default value is 0.5.

**-maxent-sigma2***S*

Use the L2 regularization constant *S*for maxent estimation. The default value is 6 for estimation and 0.5 for adaptation.

**-maxent-convert-to-arpa**

Convert the maxent model to [ngram-format(5)](http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html) before writing it out, using the algorithm by Wu (2002).

**-unk**

Build an ``open vocabulary'' LM, i.e., one that contains the unknown-word token as a regular word. The default is to remove the unknown word.

**-map-unk***word*

Map out-of-vocabulary words to *word*, rather than the default **<unk>**tag.

**-trust-totals**

Force the lower-order counts to be used as total counts in estimating N-gram probabilities. Usually these totals are recomputed from the higher-order counts.

**-prune***threshold*

Prune N-gram probabilities if their removal causes (training set) perplexity of the model to increase by less than *threshold*relative.

**-minprune***n*

Only prune N-grams of length at least *n*. The default (and minimum allowed value) is 2, i.e., only unigrams are excluded from pruning.

**-debug***level*

Set debugging output from estimated LM at *level*. Level 0 means no debugging. Debugging messages are written to stderr.

**-gt*n*min***count*

where *n*is 1, 2, 3, 4, 5, 6, 7, 8, or 9. Set the minimal count of N-grams of order *n*that will be included in the LM. All N-grams with frequency lower than that will effectively be discounted to 0. If *n*is omitted the parameter for N-grams of order > 9 is set.   
NOTE: This option affects not only the default Good-Turing discounting but the alternative discounting methods described below as well.

**-gt*n*max***count*

where *n*is 1, 2, 3, 4, 5, 6, 7, 8, or 9. Set the maximal count of N-grams of order *n*that are discounted under Good-Turing. All N-grams more frequent than that will receive maximum likelihood estimates. Discounting can be effectively disabled by setting this to 0. If *n*is omitted the parameter for N-grams of order > 9 is set.

In the following discounting parameter options, the order *n*may be omitted, in which case a default for all N-gram orders is set. The corresponding discounting method then becomes the default method for all orders, unless specifically overridden by an option with *n*. If no discounting method is specified, Good-Turing is used.

**-gt*n****gtfile*

where *n*is 1, 2, 3, 4, 5, 6, 7, 8, or 9. Save or retrieve Good-Turing parameters (cutoffs and discounting factors) in/from *gtfile*. This is useful as GT parameters should always be determined from unlimited vocabulary counts, whereas the eventual LM may use a limited vocabulary. The parameter files may also be hand-edited. If an **-lm**option is specified the GT parameters are read from *gtfile*, otherwise they are computed from the current counts and saved in *gtfile*.

**-cdiscount*n****discount*

where *n*is 1, 2, 3, 4, 5, 6, 7, 8, or 9. Use Ney's absolute discounting for N-grams of order *n*, using *discount*as the constant to subtract.

**-wbdiscount*n***

where *n*is 1, 2, 3, 4, 5, 6, 7, 8, or 9. Use Witten-Bell discounting for N-grams of order *n*. (This is the estimator where the first occurrence of each word is taken to be a sample for the ``unseen'' event.)

**-ndiscount*n***

where *n*is 1, 2, 3, 4, 5, 6, 7, 8, or 9. Use Ristad's natural discounting law for N-grams of order *n*.

**-addsmooth*n****delta*

where *n*is 1, 2, 3, 4, 5, 6, 7, 8, or 9. Smooth by adding *delta*to each N-gram count. This is usually a poor smoothing method and included mainly for instructional purposes.

**-kndiscount*n***

where *n*is 1, 2, 3, 4, 5, 6, 7, 8, or 9. Use Chen and Goodman's modified Kneser-Ney discounting for N-grams of order *n*.

**-kn-counts-modified**

Indicates that input counts have already been modified for Kneser-Ney smoothing. If this option is not given, the KN discounting method modifies counts (except those of highest order) in order to estimate the backoff distributions. When using the **-write**and related options the output will reflect the modified counts.

**-kn-modify-counts-at-end**

Modify Kneser-Ney counts after estimating discounting constants, rather than before as is the default.

**-kn*n****knfile*

where *n*is 1, 2, 3, 4, 5, 6, 7, 8, or 9. Save or retrieve Kneser-Ney parameters (cutoff and discounting constants) in/from *knfile*. This is useful as smoothing parameters should always be determined from unlimited vocabulary counts, whereas the eventual LM may use a limited vocabulary. The parameter files may also be hand-edited. If an **-lm**option is specified the KN parameters are read from *knfile*, otherwise they are computed from the current counts and saved in *knfile*.

**-ukndiscount*n***

where *n*is 1, 2, 3, 4, 5, 6, 7, 8, or 9. Use the original (unmodified) Kneser-Ney discounting method for N-grams of order *n*.

**-interpolate*n***

where *n*is 1, 2, 3, 4, 5, 6, 7, 8, or 9. Causes the discounted N-gram probability estimates at the specified order *n*to be interpolated with lower-order estimates. (The result of the interpolation is encoded as a standard backoff model and can be evaluated as such -- the interpolation happens at estimation time.) This sometimes yields better models with some smoothing methods (see Chen & Goodman, 1998). Only Witten-Bell, absolute discounting, and (original or modified) Kneser-Ney smoothing currently support interpolation.

**-meta-tag***string*

Interpret words starting with *string*as count-of-count (meta-count) tags. For example, an N-gram

a b *string*3 4

means that there were 4 trigrams starting with "a b" that occurred 3 times each. Meta-tags are only allowed in the last position of an N-gram.   
Note: when using **-tolower**the meta-tag *string*must not contain any uppercase characters.

**-read-with-mincounts**

Save memory by eliminating N-grams with counts that fall below the thresholds set by **-gt***N***min** options during **-read**operation (this assumes the input counts contain no duplicate N-grams). Also, if **-meta-tag**is defined, these low-count N-grams will be converted to count-of-count N-grams, so that smoothing methods that need this information still work correctly.

**SEE ALSO**

[ngram-merge(1)](http://www.speech.sri.com/projects/srilm/manpages/ngram-merge.1.html), [ngram(1)](http://www.speech.sri.com/projects/srilm/manpages/ngram.1.html), [ngram-class(1)](http://www.speech.sri.com/projects/srilm/manpages/ngram-class.1.html), [training-scripts(1)](http://www.speech.sri.com/projects/srilm/manpages/training-scripts.1.html), [lm-scripts(1)](http://www.speech.sri.com/projects/srilm/manpages/lm-scripts.1.html), [ngram-format(5)](http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html).   
T. Alumäe and M. Kurimo, ``Efficient Estimation of Maximum Entropy Language Models with N-gram features: an SRILM extension,'' *Proc. Interspeech*, 1820-1823, 2010.   
S. F. Chen and J. Goodman, ``An Empirical Study of Smoothing Techniques for Language Modeling,'' TR-10-98, Computer Science Group, Harvard Univ., 1998.   
S. M. Katz, ``Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer,'' *IEEE Trans. ASSP* 35(3), 400-401, 1987.   
R. Kneser and H. Ney, ``Improved backing-off for M-gram language modeling,'' *Proc. ICASSP*, 181-184, 1995.   
H. Ney and U. Essen, ``On Smoothing Techniques for Bigram-based Natural Language Modelling,'' *Proc. ICASSP*, 825-828, 1991.   
E. S. Ristad, ``A Natural Law of Succession,'' CS-TR-495-95, Comp. Sci. Dept., Princeton Univ., 1995.   
I. H. Witten and T. C. Bell, ``The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression,'' *IEEE Trans. Information Theory* 37(4), 1085-1094, 1991.   
J. Wu (2002), ``Maximum Entropy Language Modeling with Non-Local Dependencies,'' doctoral dissertation, Johns Hopkins University, 2002.

**BUGS**

Several of the LM types supported by [ngram(1)](http://www.speech.sri.com/projects/srilm/manpages/ngram.1.html) don't have explicit support in **ngram-count**. Instead, they are built by separately manipulating N-gram counts, followed by standard N-gram model estimation.   
LM support for tagged words is incomplete.   
Only absolute and Witten-Bell discounting currently support fractional counts.   
The combination of **-read-with-mincounts**and **-meta-tag**preserves enough count-of-count information for *applying*discounting parameters to the input counts, but it does not necessarily allow the parameters to be correctly*estimated*. Therefore, discounting parameters should always be estimated from full counts (e.g., using the helper[training-scripts(1)](http://www.speech.sri.com/projects/srilm/manpages/training-scripts.1.html)), and then read from files.   
Smoothing with **-kndiscount**or **-ukndiscount**has a side-effect on the N-gram counts, since the lower-order counts are destructively modified according to the KN smoothing approach (Kneser & Ney, 1995). The **-write** option will output these modified counts, and count cutoffs specified by **-gt***N***min** operate on the modified counts, potentially leading to a different set of N-grams being retained than with other discounting methods. This can be considered either a feature or a bug.

**AUTHOR**

Andreas Stolcke <stolcke@speech.sri.com>   
Tanel Alumäe <tanel.alumae@phon.ioc.ee>   
Copyright (c) 1995-2011 SRI International   
Copyright (c) 2009-2013 Tanel Alumäe   
Copyright (c) 2012-2013 Microsoft Corp.

# ngram

## NAME

ngram - apply N-gram language models

## SYNOPSIS

**ngram** [ **-help** ] *option* ...

## DESCRIPTION

**ngram**performs various operations with N-gram-based and related language models, including sentence scoring, perplexity computation, sentences generation, and various types of model interpolation. The N-gram language models are read from files in ARPA [ngram-format(5)](http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html); various extended language model formats are described with the options below.

## OPTIONS

Each filename argument can be an ASCII file, or a compressed file (name ending in .Z or .gz), or ``-'' to indicate stdin/stdout.

**-help**

Print option summary.

**-version**

Print version information.

**-order***n*

Set the maximal N-gram order to be used, by default 3. NOTE: The order of the model is not set automatically when a model file is read, so the same file can be used at various orders. To use models of order higher than 3 it is always necessary to specify this option.

**-debug***level*

Set the debugging output level (0 means no debugging output). Debugging messages are sent to stderr, with the exception of **-ppl**output as explained below.

**-memuse**

Print memory usage statistics for the LM.

The following options determine the type of LM to be used.

**-null**

Use a `null' LM as the main model (one that gives probability 1 to all words). This is useful in combination with mixture creation or for debugging.

**-use-server***S*

Use a network LM server (typically implemented by **ngram**with the **-server-port**option) as the main model. The server specification *S*can be an unsigned integer port number (referring to a server port running on the local host), a hostname (referring to default port 2525 on the named host), or a string of the form *port*@*host*, where*port*is a portnumber and *host*is either a hostname ("dukas.speech.sri.com") or IP number in dotted-quad format ("140.44.1.15").   
For server-based LMs, the **-order**option limits the context length of N-grams queried by the client (with 0 denoting unlimited length). Hence, the effective LM order is the mimimum of the client-specified value and any limit implemented in the server.   
When **-use-server**is specified, the arguments to the options **-mix-lm**, **-mix-lm2**, etc. are also interpreted as network LM server specifications provided they contain a '@' character and do not contain a '/' character. This allows the creation of mixtures of several file- and/or network-based LMs.

**-cache-served-ngrams**

Enables client-side caching of N-gram probabilities to eliminated duplicate network queries, in conjunction with **-use-server**. This results in a substantial speedup for typical tasks (especially N-best rescoring) but requires memory in the client that may grow linearly with the amount of data processed.

**-lm***file*

Read the (main) N-gram model from *file*. This option is always required, unless **-null**was chosen. Unless modified by other options, the *file*is assumed to contain an N-gram backoff language model in [ngram-format(5)](http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html).

**-tagged**

Interpret the LM as containing word/tag N-grams.

**-skip**

Interpret the LM as a ``skip'' N-gram model.

**-hidden-vocab***file*

Interpret the LM as an N-gram model containing hidden events between words. The list of hidden event tags is read from *file*.   
Hidden event definitions may also follow the N-gram definitions in the LM file (the argument to **-lm**). The format for such definitions is

*event* [**-delete** *D*] [**-repeat** *R*] [**-insert** *w*] [**-observed**] [**-omit**]

The optional flags after the event name modify the default behavior of hidden events in the model. By default events are unobserved pseudo-words of which at most one can occur between regular words, and which are added to the context to predict following words and events. (A typical use would be to model hidden sentence boundaries.) **-delete**indicates that upon encountering the event, *D*words are deleted from the next word's context. **-repeat**indicates that after the event the next *R*words from the context are to be repeated. **-insert**specifies that an (unobserved) word *w*is to be inserted into the history. **-observed**specifies the event tag is not hidden, but observed in the word stream. **-omit**indicates that the event tag itself is not to be added to the history for predicting the following words.   
The hidden event mechanism represents a generalization of the disfluency LM enabled by **-df**.

**-hidden-not**

Modifies processing of hidden event N-grams for the case that the event tags are embedded in the word stream, as opposed to inferred through dynamic programming.

**-df**

Interpret the LM as containing disfluency events. This enables an older form of hidden-event LM used in Stolcke & Shriberg (1996). It is roughly equivalent to a hidden-event LM with

UH -observed -omit (filled pause)

UM -observed -omit (filled pause)

@SDEL -insert <s> (sentence restart)

@DEL1 -delete 1 -omit (1-word deletion)

@DEL2 -delete 2 -omit (2-word deletion)

@REP1 -repeat 1 -omit (1-word repetition)

@REP2 -repeat 2 -omit (2-word repetition)

**-classes***file*

Interpret the LM as an N-gram over word classes. The expansions of the classes are given in *file* in [classes-format(5)](http://www.speech.sri.com/projects/srilm/manpages/classes-format.5.html). Tokens in the LM that are not defined as classes in *file*are assumed to be plain words, so that the LM can contain mixed N-grams over both words and word classes.   
Class definitions may also follow the N-gram definitions in the LM file (the argument to **-lm**). In that case **-classes /dev/null** should be specified to trigger interpretation of the LM as a class-based model. Otherwise, class definitions specified with this option override any definitions found in the LM file itself.

**-simple-classes**

Assume a "simple" class model: each word is member of at most one word class, and class expansions are exactly one word long.

**-expand-classes***k*

Replace the read class-N-gram model with an (approximately) equivalent word-based N-gram. The argument *k*limits the length of the N-grams included in the new model (*k*=0 allows N-grams of arbitrary length).

**-expand-exact***k*

Use a more exact (but also more expensive) algorithm to compute the conditional probabilities of N-grams expanded from classes, for N-grams of length *k*or longer (*k*=0 is a special case and the default, it disables the exact algorithm for all N-grams). The exact algorithm is recommended for class-N-gram models that contain multi-word class expansions, for N-gram lengths exceeding the order of the underlying class N-grams.

**-codebook***file*

Read a codebook for quantized log probabilies from *file*. The parameters in an N-gram language model file specified by **-lm**are then assumed to represent codebook indices instead of log probabilities.

**-decipher**

Use the N-gram model exactly as the Decipher(TM) recognizer would, i.e., choosing the backoff path if it has a higher probability than the bigram transition, and rounding log probabilities to bytelog precision.

**-factored**

Use a factored N-gram model, i.e., a model that represents words as vectors of feature-value pairs and models sequences of words by a set of conditional dependency relations between factors. Individual dependencies are modeled by standard N-gram LMs, allowing however for a generalized backoff mechanism to combine multiple backoff paths (Bilmes and Kirchhoff 2003). The **-lm**, **-mix-lm**, etc. options name FLM specification files in the format described in Kirchhoff et al. (2002).

**-hmm**

Use an HMM of N-grams language model. The **-lm**option specifies a file that describes a probabilistic graph, with each line corresponding to a node or state. A line has the format:

*statename* *ngram-file* *s1* *p1* *s2* *p2* ...

where *statename*is a string identifying the state, *ngram-file*names a file containing a backoff N-gram model,*s1*,*s2*, ... are names of follow-states, and *p1*,*p2*, ... are the associated transition probabilities. A filename of ``-'' can be used to indicate the N-gram model data is included in the HMM file, after the current line. (Further HMM states may be specified after the N-gram data.)   
The names **INITIAL**and **FINAL**denote the start and end states, respectively, and have no associated N-gram model (*ngram-file*must be specified as ``.'' for these). The **-order**option specifies the maximal N-gram length in the component models.   
The semantics of an HMM of N-grams is as follows: as each state is visited, words are emitted from the associated N-gram model. The first state (corresponding to the start-of-sentence) is **INITIAL**. A state is left with the probability of the end-of-sentence token in the respective model, and the next state is chosen according to the state transition probabilities. Each state has to emit at least one word. The actual end-of-sentence is emitted if and only if the **FINAL**state is reached. Each word probability is conditioned on all preceding words, regardless of whether they were emitted in the same or a previous state.

**-count-lm**

Use a count-based interpolated LM. The **-lm**option specifies a file that describes a set of N-gram counts along with interpolation weights, based on which Jelinek-Mercer smoothing in the formulation of Chen and Goodman (1998) is performed. The file format is

**order** *N*

**vocabsize** *V*

**totalcount** *C*

**mixweights** *M*

*w01* *w02* ... *w0N*

*w11* *w12* ... *w1N*

...

*wM1* *wM2* ... *wMN*

**countmodulus** *m*

**google-counts** *dir*

**counts** *file*

Here *N*is the model order (maximal N-gram length), although as with backoff models, the actual value used is overridden by the **-order**command line when the model is read in. *V*gives the vocabulary size and *C*the sum of all unigram counts. *M*specifies the number of mixture weight bins (minus 1). *m*is the width of a mixture weight bin. Thus, *wij*is the mixture weight used to interpolate an *j*-th order maximum-likelihood estimate with lower-order estimates given that the (*j*-1)-gram context has been seen with a frequency between *i*\**m* and (*i*+1)\**m*-1times. (For contexts with frequency greater than *M*\**m*, the *i*=*M* weights are used.) The N-gram counts themselves are given in an indexed directory structure rooted at *dir*, in an external *file*, or, if *file*is the string **-**,starting on the line following the **counts**keyword.

**-msweb-lm**

Use a Microsoft Web N-gram language model. The **-lm**option specifies a file that contains the parameters for retrieving N-gram probabilities from the service described at <http://web-ngram.research.microsoft.com/> and in Gao et al. (2010). The **-cache-served-ngrams**option applies, and causes N-gram probabilities retrieved from the server to be stored for later reuse. The file format expected by **-lm**is as follows, with default values listed after each parameter name:

**servername** web-ngram.research.microsoft.com

**serverport** 80

**urlprefix** /rest/lookup.svc

**usertoken** *xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx*

**catalog** bing-body

**version** jun09

**modelorder** *N*

**cacheorder** 0 (*N* with **-cache-served-ngrams**)

**maxretries** 2

The string following **usertoken**is obligatory and is a user-specific key that must be obtained by emailing <webngram@microsoft.com>. The language model order *N*defaults to the value of the **-order** option. It is recommended that **modelorder**be specified in case the **-order** argument exceeds the server's model order. Note also that the LM thus created will have no predefined vocabulary. Any operations that rely on the vocabulary being known (such as sentence generation) will require one to be specified explicitly with **-vocab**.

**-maxent**

Read a maximum entropy N-gram model. The model file is specified by **-lm**.

**-mix-maxent**

Indicates that all mixture model components specified by **-mix-lm**and related options are maxent models. Without this option, an interpolation of a single maxent model (specified by **-lm**) with standard backoff models (specified by **-mix-lm**etc.) is performed. The option **-bayes***N* should also be given, unless used in combination with **-maxent-convert-to-arpa**(see below).

**-maxent-convert-to-arpa**

Indicates that the **-lm**option specifies a maxent model file, but that the model is to be converted to a backoff model using the algorithm by Wu (2002). This option also triggers conversion of maxent models used with **-mix-maxent**.

**-vocab***file*

Initialize the vocabulary for the LM from *file*. This is especially useful if the LM itself does not specify a complete vocabulary, e.g., as with **-null**.

**-vocab-aliases***file*

Reads vocabulary alias definitions from *file*, consisting of lines of the form

*alias* *word*

This causes all tokens *alias*to be mapped to *word*.

**-nonevents***file*

Read a list of words from *file*that are to be considered non-events, i.e., that should only occur in LM contexts, but not as predictions. Such words are excluded from sentence generation (**-gen**) and probability summation (**-ppl -debug 3**).

**-limit-vocab**

Discard LM parameters on reading that do not pertain to the words specified in the vocabulary. The default is that words used in the LM are automatically added to the vocabulary. This option can be used to reduce the memory requirements for large LMs that are going to be evaluated only on a small vocabulary subset.

**-unk**

Indicates that the LM contains the unknown word, i.e., is an open-class LM.

**-map-unk***word*

Map out-of-vocabulary words to *word*, rather than the default **<unk>**tag.

**-tolower**

Map all vocabulary to lowercase. Useful if case conventions for text/counts and language model differ.

**-multiwords**

Split input words consisting of multiwords joined by underscores into their components, before evaluating LM probabilities.

**-multi-char***C*

Character used to delimit component words in multiwords (an underscore character by default).

**-zeroprob-word***W*

If a word token is assigned a probability of zero by the LM, look up the word *W*instead. This is useful to avoid zero probabilities when processing input with an LM that is mismatched in vocabulary.

**-mix-lm***file*

Read a second N-gram model for interpolation purposes. The second and any additional interpolated models can also be class N-grams (using the same **-classes**definitions), but are otherwise constrained to be standard N-grams, i.e., the options **-df**, **-tagged**, **-skip**, and **-hidden-vocab**do not apply to them.   
**NOTE:**Unless **-bayes**(see below) is specified, **-mix-lm**triggers a static interpolation of the models in memory. In most cases a more efficient, dynamic interpolation is sufficient, requested by **-bayes 0**. Also, mixing models of different type (e.g., word-based and class-based) will *only*work correctly with dynamic interpolation.

**-lambda***weight*

Set the weight of the main model when interpolating with **-mix-lm**. Default value is 0.5.

**-mix-lm2***file*

**-mix-lm3***file*

**-mix-lm4***file*

**-mix-lm5***file*

**-mix-lm6***file*

**-mix-lm7***file*

**-mix-lm8***file*

**-mix-lm9***file*

Up to 9 more N-gram models can be specified for interpolation.

**-mix-lambda2***weight*

**-mix-lambda3***weight*

**-mix-lambda4***weight*

**-mix-lambda5***weight*

**-mix-lambda6***weight*

**-mix-lambda7***weight*

**-mix-lambda8***weight*

**-mix-lambda9***weight*

These are the weights for the additional mixture components, corresponding to **-mix-lm2**through **-mix-lm9**. The weight for the **-mix-lm**model is 1 minus the sum of **-lambda**and **-mix-lambda2**through **-mix-lambda9**.

**-loglinear-mix**

Implement a log-linear (rather than linear) mixture LM, using the parameters above.

**-context-priors** file

Read context-dependent mixture weight priors from *file*. Each line in *file*should contain a context N-gram (most recent word first) followed by a vector of mixture weights whose length matches the number of LMs being interpolated. (This and the following options currently only apply to linear interpolation.)

**-bayes***length*

Interpolate models using posterior probabilities based on the likelihoods of local N-gram contexts of length*length*. The **-lambda**values are used as prior mixture weights in this case. This option can also be combined with **-context-priors**, in which case the *length*parameter also controls how many words of context are maximally used to look up mixture weights. If **-context-priors** is used without **-bayes**, the context length used is set by the **-order**option and a merged (statically interpolated) N-gram model is created.

**-bayes-scale***scale*

Set the exponential scale factor on the context likelihoods in conjunction with the **-bayes**function. Default value is 1.0.

**-read-mix-lms**

Read a list of linearly interpolated (mixture) LMs and their weights from the *file*specified with **-lm**, instead of gathering this information from the command line options above. Each line in *file*starts with the filename containing the component LM, followed by zero or more component-specific options:

**-weight***W*

the prior weight given to the component LM

**-order***N*

the maximal ngram order to use

**-type***T*

the LM type, one of **ARPA**(the default), **COUNTLM**, **MAXENT**, **LMCLIENT**, or **MSWEBLM**

**-classes***C*

the word class definitions for the component LM (which must be of type ARPA)

**-cache-served-ngrams**

enables client-side caching for LMs of type LMCLIENT or MSWEBLM.

The global options **-bayes**, **-bayes-scale**, and **-context-priors**still apply with **-read-mix-lms**. When **-bayes** is NOT used, the interpolation is static by ngram merging, and forces all component LMs to be of type ARPA or MAXENT.

**-cache***length*

Interpolate the main LM (or the one resulting from operations above) with a unigram cache language model based on a history of *length*words.

**-cache-lambda***weight*

Set interpolation weight for the cache LM. Default value is 0.05.

**-dynamic**

Interpolate the main LM (or the one resulting from operations above) with a dynamically changing LM. LM changes are indicated by the tag ``<LMstate>'' starting a line in the input to **-ppl**, **-counts**, or **-rescore**, followed by a filename containing the new LM.

**-dynamic-lambda***weight*

Set interpolation weight for the dynamic LM. Default value is 0.05.

**-adapt-marginals***LM*

Use an LM obtained by adapting the unigram marginals to the values specified in the *LM*in [ngram-format(5)](http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html), using the method described in Kneser et al. (1997). The LM to be adapted is that constructed according to the other options.

**-base-marginals***LM*

Specify the baseline unigram marginals in a separate file *LM*, which must be in [ngram-format(5)](http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html) as well. If not specified, the baseline marginals are taken from the model to be adapted, but this might not be desirable, e.g., when Kneser-Ney smoothing was used.

**-adapt-marginals-beta***B*

The exponential weight given to the ratio between adapted and baseline marginals. The default is 0.5.

**-adapt-marginals-ratios**

Compute and output only the log ratio between the adapted and the baseline LM probabilities. These can be useful as a separate knowledge source in N-best rescoring.

The following options specify the operations performed on/with the LM constructed as per the options above.

**-renorm**

Renormalize the main model by recomputing backoff weights for the given probabilities.

**-prune***threshold*

Prune N-gram probabilities if their removal causes (training set) perplexity of the model to increase by less than *threshold*relative.

**-prune-history-lm***L*

Read a separate LM from file *L*and use it to obtain the history marginal probabilities required for computing the entropy loss incurred by pruning an N-gram. The LM needs to only be of an order one less than the LM being pruned. If this option is not used the LM being pruned is used to compute history marginals. This option is useful because, as pointed out by Chelba et al. (2010), the lower-order N-gram probabilities in Kneser-Ney smoothed LMs are unsuitable for this purpose.

**-prune-lowprobs**

Prune N-gram probabilities that are lower than the corresponding backed-off estimates. This generates N-gram models that can be correctly converted into probabilistic finite-state networks.

**-minprune***n*

Only prune N-grams of length at least *n*. The default (and minimum allowed value) is 2, i.e., only unigrams are excluded from pruning. This option applies to both **-prune**and **-prune-lowprobs**.

**-rescore-ngram***file*

Read an N-gram LM from *file*and recompute its N-gram probabilities using the LM specified by the other options; then renormalize and evaluate the resulting new N-gram LM.

**-write-lm***file*

Write a model back to *file*. The output will be in the same format as read by **-lm**, except if operations such as**-mix-lm**or **-expand-classes**were applied, in which case the output will contain the generated single N-gram backoff model in ARPA [ngram-format(5)](http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html).

**-write-bin-lm***file*

Write a model to *file*using a binary data format. This is only supported by certain model types, specifically, those based on N-gram backoff models and N-gram counts. Binary model files are recognized automatically by the**-read**function. If an LM class does not provide a binary format the default (text) format will be output instead.

**-write-vocab***file*

Write the LM's vocabulary to *file*.

**-gen***number*

Generate *number*random sentences from the LM.

**-gen-prefixes***file*

Read a list of sentence prefixes from *file*and generate random word strings conditioned on them, one per line. (Note: The start-of-sentence tag **<s>**is not automatically added to these prefixes.)

**-seed***value*

Initialize the random number generator used for sentence generation using seed *value*. The default is to use a seed that should be close to unique for each invocation of the program.

**-ppl***textfile*

Compute sentence scores (log probabilities) and perplexities from the sentences in *textfile*, which should contain one sentence per line. The **-debug**option controls the level of detail printed, even though output is to stdout (not stderr).

**-debug 0**

Only summary statistics for the entire corpus are printed, as well as partial statistics for each input portion delimited by escaped lines (see **-escape**). These statistics include the number of sentences, words, out-of-vocabulary words and zero-probability tokens in the input, as well as its total log probability and perplexity. Perplexity is given with two different normalizations: counting all input tokens (``ppl'') and excluding end-of-sentence tags (``ppl1'').

**-debug 1**

Statistics for individual sentences are printed.

**-debug 2**

Probabilities for each word, plus LM-dependent details about backoff used etc., are printed.

**-debug 3**

Probabilities for all words are summed in each context, and the sum is printed. If this differs significantly from 1, a warning message to stderr will be issued.

**-debug 4**

Outputs ranking statistics (number of times the actual word's probability was ranked in top 1, 5, 10 among all possible words, both excluding and including end-of-sentence tokens), as well as quadratic and absolute loss averages (based on how much actual word probability differs from 1).

**-nbest***file*

Read an N-best list in [nbest-format(5)](http://www.speech.sri.com/projects/srilm/manpages/nbest-format.5.html) and rerank the hypotheses using the specified LM. The reordered N-best list is written to stdout. If the N-best list is given in ``NBestList1.0'' format and contains composite acoustic/language model scores, then **-decipher-lm**and the recognizer language model and word transition weights (see below) need to be specified so the original acoustic scores can be recovered.

**-nbest-files***filelist*

Process multiple N-best lists whose filenames are listed in *filelist*.

**-write-nbest-dir***dir*

Deposit rescored N-best lists into directory *dir*, using filenames derived from the input ones.

**-decipher-nbest**

Output rescored N-best lists in Decipher 1.0 format, rather than SRILM format.

**-no-reorder**

Output rescored N-best lists without sorting the hypotheses by their new combined scores.

**-split-multiwords**

Split multiwords into their components when reading N-best lists; the rescored N-best lists thus no longer contain multiwords. (Note this is different from the **-multiwords**option, which leaves the input word stream unchanged and splits multiwords only for the purpose of LM probability computation.)

**-max-nbest***n*

Limits the number of hypotheses read from an N-best list. Only the first *n*hypotheses are processed.

**-rescore***file*

Similar to **-nbest**, but the input is processed as a stream of N-best hypotheses (without header). The output consists of the rescored hypotheses in SRILM format (the third of the formats described in [nbest-format(5)](http://www.speech.sri.com/projects/srilm/manpages/nbest-format.5.html)).

**-decipher-lm***model-file*

Designates the N-gram backoff model (typically a bigram) that was used by the Decipher(TM) recognizer in computing composite scores for the hypotheses fed to **-rescore**or **-nbest**. Used to compute acoustic scores from the composite scores.

**-decipher-order***N*

Specifies the order of the Decipher N-gram model used (default is 2).

**-decipher-nobackoff**

Indicates that the Decipher N-gram model does not contain backoff nodes, i.e., all recognizer LM scores are correct up to rounding.

**-decipher-lmw***weight*

Specifies the language model weight used by the recognizer. Used to compute acoustic scores from the composite scores.

**-decipher-wtw***weight*

Specifies the word transition weight used by the recognizer. Used to compute acoustic scores from the composite scores.

**-escape***string*

Set an ``escape string'' for the **-ppl**, **-counts**, and **-rescore**computations. Input lines starting with *string*are not processed as sentences and passed unchanged to stdout instead. This allows associated information to be passed to scoring scripts etc.

**-counts***countsfile*

Perform a computation similar to **-ppl**, but based only on the N-gram counts found in *countsfile*. Probabilities are computed for the last word of each N-gram, using the other words as contexts, and scaling by the associated N-gram count. Summary statistics are output at the end, as well as before each escaped input line if **-debug**level 1 or higher is set.

**-count-order***n*

Use only counts up to order *n*in the **-counts**computation. The default value is the order of the LM (the value specified by **-order**).

**-float-counts**

Allow processing of fractional counts with **-counts**.

**-counts-entropy**

Weight the log probabilities for **-counts**processing by the join probabilities of the N-grams. This effectively computes the sum over p(w,h) log p(w|h), i.e., the entropy of the model. In debugging mode, both the conditional log probabilities and the corresponding joint probabilities are output.

**-server-port***P*

Start a network server that listens on port *P*and returns N-gram probabilities. The server will write a one-line "ready" message and then read N-grams, one per line. For each N-gram, a conditional log probability is computed as specified by other options, and written back to the client (in text format). The server will continue accepting connections until killed by an external signal.

**-server-maxclients***M*

Limits the number of simultaneous connections accepted by the network LM server to *M*. Once the limit is reached, additional connection requests (e.g., via **ngram** **-use-server**) will hang until another client terminates its connection.

**-skipoovs**

Instruct the LM to skip over contexts that contain out-of-vocabulary words, instead of using a backoff strategy in these cases.

**-noise***noise-tag*

Designate *noise-tag*as a vocabulary item that is to be ignored by the LM. (This is typically used to identify a noise marker.) Note that the LM specified by **-decipher-lm**does NOT ignore this *noise-tag*since the DECIPHER recognizer treats noise as a regular word.

**-noise-vocab***file*

Read several noise tags from *file*, instead of, or in addition to, the single noise tag specified by **-noise**.

**-reverse**

Reverse the words in a sentence for LM scoring purposes. (This assumes the LM used is a ``right-to-left'' model.) Note that the LM specified by **-decipher-lm**is always applied to the original, left-to-right word sequence.

**-no-sos**

Disable the automatic insertion of start-of-sentence tokens for sentence probability computation. The probability of the initial word is thus computed with an empty context.

**-no-eos**

Disable the automatic insertion of end-of-sentence tokens for sentence probability computation. End-of-sentence is thus excluded from the total probability.

## SEE ALSO

[ngram-count(1)](http://www.speech.sri.com/projects/srilm/manpages/ngram-count.1.html), [ngram-class(1)](http://www.speech.sri.com/projects/srilm/manpages/ngram-class.1.html), [lm-scripts(1)](http://www.speech.sri.com/projects/srilm/manpages/lm-scripts.1.html), [ppl-scripts(1)](http://www.speech.sri.com/projects/srilm/manpages/ppl-scripts.1.html), [pfsg-scripts(1)](http://www.speech.sri.com/projects/srilm/manpages/pfsg-scripts.1.html), [nbest-scripts(1)](http://www.speech.sri.com/projects/srilm/manpages/nbest-scripts.1.html), [ngram-format(5)](http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html),[nbest-format(5)](http://www.speech.sri.com/projects/srilm/manpages/nbest-format.5.html), [classes-format(5)](http://www.speech.sri.com/projects/srilm/manpages/classes-format.5.html).   
J. A. Bilmes and K. Kirchhoff, ``Factored Language Models and Generalized Parallel Backoff,'' *Proc. HLT-NAACL*, pp. 4-6, Edmonton, Alberta, 2003.   
C. Chelba, T. Brants, W. Neveitt, and P. Xu, ``Study on Interaction Between Entropy Pruning and Kneser-Ney Smoothing,'' *Proc. Interspeech*, pp. 2422-2425, Makuhari, Japan, 2010.   
S. F. Chen and J. Goodman, ``An Empirical Study of Smoothing Techniques for Language Modeling,'' TR-10-98, Computer Science Group, Harvard Univ., 1998.   
J. Gao, P. Nguyen, X. Li, C. Thrasher, M. Li, and K. Wang, ``A Comparative Study of Bing Web N-gram Language Models for Web Search and Natural Language Processing,'' Proc. SIGIR, July 2010.   
K. Kirchhoff et al., ``Novel Speech Recognition Models for Arabic,'' Johns Hopkins University Summer Research Workshop 2002, Final Report.   
R. Kneser, J. Peters and D. Klakow, ``Language Model Adaptation Using Dynamic Marginals'', *Proc. Eurospeech*, pp. 1971-1974, Rhodes, 1997.   
A. Stolcke and E. Shriberg, ``Statistical language modeling for speech disfluencies,'' Proc. IEEE ICASSP, pp. 405-409, Atlanta, GA, 1996.   
A. Stolcke,`` Entropy-based Pruning of Backoff Language Models,'' *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pp. 270-274, Lansdowne, VA, 1998.   
A. Stolcke et al., ``Automatic Detection of Sentence Boundaries and Disfluencies based on Recognized Words,'' *Proc. ICSLP*, pp. 2247-2250, Sydney, 1998.   
M. Weintraub et al., ``Fast Training and Portability,'' in Research Note No. 1, Center for Language and Speech Processing, Johns Hopkins University, Baltimore, Feb. 1996.   
J. Wu (2002), ``Maximum Entropy Language Modeling with Non-Local Dependencies,'' doctoral dissertation, Johns Hopkins University, 2002.

## BUGS

Some LM types (such as Bayes-interpolated and factored LMs) currently do not support the **-write-lm**function.

For the **-limit-vocab**option to work correctly with hidden event and class N-gram LMs, the event/class vocabularies have to be specified by options (**-hidden-vocab**and **-classes**, respectively). Embedding event/class definitions in the LM file only will not work correctly.

Sentence generation is slow and takes time proportional to the vocabulary size.

The file given by **-classes**is read multiple times if **-limit-vocab**is in effect or if a mixture of LMs is specified. This will lead to incorrect behavior if the argument of **-classes**is stdin (``-'').

Also, **-limit-vocab**will not work correctly with LM operations that require the entire vocabulary to be enumerated, such as **-adapt-marginals**or perplexity computation with **-debug 3**.

The **-multiword**option implicitly adds all word strings to the vocabulary. Therefore, no OOVs are reported, only zero probability words.

Operations that require enumeration of the entire LM vocabulary will not currently work with **-use-server**, since the client side only has knowledge of words it has already processed. This affects the **-gen**and **-adapt-marginals**options, as well as **-ppl**with **-debug 3**. A workaround is to specify the complete vocabulary with **-vocab**on the client side.

The reading of quantized LM parameters with the **-codebook**option is currently only supported for N-gram LMs in[ngram-format(5)](http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html).

## AUTHORS

Andreas Stolcke <andreas.stolcke@microsoft.com>   
Jing Zheng <zj@speech.sri.com>   
Tanel Alumae <tanel.alumae@phon.ioc.ee>   
Copyright (c) 1995-2012 SRI International   
Copyright (c) 2009-2013 Tanel Alumae   
Copyright (c) 2012-2013 Microsoft Corp.