

The Morphological Component of a Joint
Morphological-Distributional Class Language
Model

Thomas Müller

May 16, 2011

Abstract

Modeling of out-of-vocabulary (OOV) words, i.e. words that do not occur in the training corpus but in the natural language processing (NLP) task at hand, is a challenging problem of statistical language modeling. We empirically investigate the relation between word context, word class and word morphology in English and present a class-based language model, which groups rare words of similar morphology together. The model improves the prediction of words after histories containing out-of-vocabulary words. The morphological features used are obtained without the use of labeled data, but produce a number of syntactically and even semantically related clusters. The overall perplexity improvement achieved by our model is 4% compared to a state of the art Kneser-Ney model and 81% on unknown histories. We conclude that the usage of morphological features in English language modeling is worthwhile.

Contents

1	Introduction	10
2	Related Work	13
3	Feature Design and Clustering	16
3.1	Morphological Features	16
3.1.1	The Reports Algorithm	16
3.1.2	Affixation-based Features	18
3.1.3	Stem-based Features	19
3.2	Shape-like Features	19
3.2.1	Capitalization	19
3.2.2	Special Characters	20
3.2.3	Word Length	21
3.3	K-Means Clustering	21
3.4	Evaluation Criteria	24
3.4.1	Pair-based Criteria	25
3.4.2	Information-Theory-based Criteria	27
3.4.3	Limitations of Contingency-Table-based Criteria	31
3.5	Experimental Setup and Results	32
4	A Morphological Language Model	35
4.1	Foundations of Language Modeling	35
4.1.1	The Maximum Likelihood Estimate	36
4.1.2	Kneser-Ney Discounting	36
4.1.3	Class-based Models	38
4.1.4	Linear Interpolation	38
4.1.5	Language Model Evaluation	41
4.2	Design of a Morphological Language Model	41
4.2.1	A Preliminary Model	41
4.2.2	Evaluation of the Preliminary Model	43
4.2.3	The Final Model	44

4.2.4	Evaluation of the Final Model	46
5	Conclusion	55
6	Future Work	56
	Appendix – Proof $NI = V_1$	57
	Appendix – The Penn Treebank Tagset	58

List of Figures

3.1	Example of the transitional probabilities in the word refundings indicating “-ing” and “-s” as good suffix candidates. . .	17
3.2	An hierarchical clustering of the training set of section 4. The edge weights correspond to the number of word types of the cluster the edge is pointing to.	23
3.3	The three uppermost layers of the same clustering as in 3.2. .	23
3.4	Subtree of the right uppermost child node of the graph shown in 3.2.	24
3.5	An example demonstrating that the NVI can be greater than 1 for decent clusterings.	30
4.1	Implementation of Bahl’s algorithm. The parameter ϵ is the desired maximal distance to the true optimum.	40
4.2	A block diagram of the training procedure of the final interpolated model. Gray nodes are part of the cluster process. . .	45

List of Tables

3.1	Performance of the Length feature.	32
3.2	Performance of the feature P_1	33
3.3	Performance of the stem feature S_2	33
3.4	Performance of the stem feature P_2	33
4.1	Segmentation of the WSJ corpus	43
4.2	Perplexity results for the preliminary model.	43
4.3	Proportion of dominant POS for types with training set frequencies $f \in \{0,1\}$ and for tokens. V^* consists of all verb POS tags.	44
4.4	Perplexities over all events of valtst.	46
4.5	Perplexities over all events of the test set.	47
4.6	Perplexities over all events of valtst, where w_1 or w_2 are unknown.	48
4.7	Perplexities over all events of the test set, where w_1 or w_2 are unknown.	48
4.8	Perplexities over histories of valtst that are continued by OOV words for different thresholds.	49
4.9	Perplexities over histories of tst that are continued by OOV words for different thresholds.	49
4.10	List of the 20 best POS-classes for the c_{POS} model at $\theta = 1000$	51
4.11	The 20 best clusters of the c_∞ model at $\theta = 1000$, using only the suffix features.	52
4.12	List of more common suffixes for the same model as in 4.11.	53
4.13	The 20 best clusters of the c_∞ model at $\theta = 1000$, using the entire feature vector.	54

Chapter 1

Introduction

Statistical language models are an important part of modern solutions to language processing problems, such as statistical machine translation, speech recognition and handwritten text recognition. Language modeling is tightly related to the question of how much text of a certain language is required to learn and reproduce its general structure. First statistical models were merely able to create sentences that consisted at least in parts of events that were seen before, i.e. occurred in the training text. In 1992, Brown et al. [4] proposed a system that grouped words of similar contexts and treated them equally. This approach is called class-based language modeling and follows the assumption that small differences between those words are noise in the training text, which should be ignored by the model. In a version of such a system [25], only the left half of the word contexts is used to create this word clustering.

In this work, we present a model that could complement the half-context model under the usage of morphological information. Our interest is both of theoretical and practical nature. The theoretical question is, how strongly related are morphology and word context and if the morphological information can be used to improve the performance of a language model by building contextually related classes. In particular it is investigated, if this can be done, even if the morphological decomposition is learned and performed completely unsupervised and thus is not nearly as accurate as the analysis of a human expert. However, not all our features are morphological. Shape-like features as e.g. capitalization and the occurrence of special characters are also used.

The practical problem we attempt to solve is related to words that appear in the language processing task at hand, but not in the training set, so called out-of-vocabulary (OOV) words. Especially for productive languages it is often necessary to at least reduce the number of OOVs. Previous work on morphological classes in English has not been able to show noticeable im-

provements in language model performance. In this work, class models based on the ideas of Brown et al. are used to tackle the problem. Context-based class models cannot naturally integrate OOVs, as their context is unknown and for this reason they cannot be assigned to the word classes. We propose a pure morphological clustering, which can be easily extended to unknown words.

A critical remaining question is, what should be considered a good clustering? Of course, the best clustering is the one that – integrated in a class-based language model – yields the highest performance in a language processing task. However, this is a very indirect method, as this performance depends on other components and parameters of the language model and the entire system. Furthermore, it is an unfeasible way of evaluating the cluster quality.

From the point of view of linguistics one could say that the goal of a language model is to produce syntactically and semantically correct sentences. Traditionally, words are assigned to word classes (part-of-speech (POS) classes) which are then used as tokens in formal grammars. So by assigning words to classes, such as verbs, nouns, adjectives and so on, one represents this *syntactical* concept in the language model. But is grouping the words corresponding to their POS classes the most promising approach? No, it is not. We already said that the *semantical* correctness of a sentence is also important. That is, grouping e.g. all past participles into one cluster is an over-generalization, as most verbs only occur with a small number of nouns. However, we have to keep in mind what kind of information we have available. And we assume that shape and morphology are mostly syntax-related. A further aspect is that we evaluate our model on a big corpus with a high number of OOV words. It is possible to accurately POS-tag such a corpus, but there are no precise tools available to do a semantic classification by word. Thus, throughout the feature design we use the similarity to POS-classes as an indicator of cluster quality. However, it is important to point out that unsupervised POS clustering using only morphological and no form of contextual information is still a virtually impossible task. There is no way to learn that “the” and “a” are similar or that “kid” and “kids” share a similar relation as “child” and “children”. So we have to assume that for a large number of the word types of a corpus no correct clustering will be achieved.

In chapter 3 we develop a number of morphological and shape-like features, which we later use to calculate word clusterings and evaluate by comparison with POS classes. As cluster evaluation is an important and recently controversially discussed topic [17,21,22] we also compare and discuss a number of older and newer cluster criteria.

In chapter 4, we use the generated morphological classes to define and evaluate a morphological language model and discuss particularly the performance on OOV words. The rest of this diploma thesis is structured as follows: in chapter 2 we discuss other morphological language models that can be found in the related literature. Chapter 5 summarizes our conclusions and in chapter 6 we discuss some ideas for future research.

Chapter 2

Related Work

A group of morphological language models presented in recent papers can be summarized under the name of factored language models [3]. Factored language models replace words by sequences of factors or features and apply statistical language modeling to these feature sequences. Especially for highly inflecting language as Finnish [7], Turkish [7, 30] and Arabic [7, 28] or compounding languages like German [2] this principle has been applied to reduce the number of out-of-vocabulary words by replacing words by sequences of morphemes. Usually the order of the used n-grams has to be increased to compensate the larger sequences, which increases the number of parameters of the model. However, logistical knowledge can be used to reduce the number of dependencies and parameters. Ghaoui et al. [10] for example divide every word w into a root r and a derivation rule g . This procedure leads to a statistical model of the form¹:

$$\begin{aligned} P(w_n|w_1 \dots w_{n-1}) &= P((r_n, g_n)|(r_1, g_1) \dots (r_{n-1}, g_{n-1})) \\ &= P(g_n|r_1 g_1 \dots r_{n-1} g_{n-1} r_n) \cdot P(r_n|r_1 g_1 \dots r_{n-1} g_{n-1}). \end{aligned}$$

They then apply two constraints to drastically reduce the number of parameters, namely that the root r_n is independent of the preceding rules and that the derivation rule g_n only depends on the types T of the preceding roots. By strictly dividing semantic and syntactic concepts, the statistical model can be approximated as:

$$P(w_n|w_1 \dots w_{n-1}) \approx P(g_n|T(r_1)g_1 \dots T(r_{n-1})g_{n-1}r_n) \cdot P(r_n|r_1 \dots r_{n-1}).$$

This even reduces the amount of parameters in comparison with a word-based model.

¹This chapter requires knowledge of the basics of language modeling. A short introduction is given in section 4.1.

Other factored language models use improved back-off techniques like the generalized parallel back-off (GPB) [3] to process the feature sequences more effectively. A standard back-off model uses a temporal back-off order. Consider a trigram model $P(w_3|w_1w_2)$ the temporal back-off sequences for this model is $P(w_3|w_1w_2)$, $P(w_3|w_2)$, $P(w_3)$. That means if the frequency of occurrences of $w_1 w_2 w_3$ is below a certain threshold τ , the model backs off to the sequence $w_2 w_3$ and potentially to the unigram count of w_3 . However, as in a factored model different features are generated from the same word there is no such natural order. Therefore, a GPB model follows and evaluates different back-off paths and chooses a selection of them to calculate the actual probability. A so called back-off function estimates the best path $l_i = l_1 \cdots l_n$. Bilmes and Kirchhoff present 32 different functions to perform a GPB of a trigram model $P(f|f_1, f_2, f_3)$. Two examples are: the path yielding the highest probability (a rather ad-hock approach not further explained by the authors):

$$g_1(f_1, f_2, f_3, f) = \operatorname{argmax}_{(m_1, m_2) \in \{(1,2), (1,3), (1,2)\}} P_{GBO}(f|f_{l_1}, f_{l_2})$$

and the path with the highest ratio between the count of the specific event and the number of possibilities to continue the back-off path:

$$g_2(f_1, f_2, f_3, f) = \operatorname{argmax}_{(m_1, m_2) \in \{(1,2), (1,3), (1,2)\}} \frac{C(f_{m_1} f_{m_2} f)}{|\{f' | C(f_{m_1} f_{m_2} f') > 0\}|}$$

The last being related to the Kneser-Ney approach as described in section 4.1.2. Of course, some adequate normalization parameter has to be found to yield a consistent statistical model. However, any further discussion of the topic lies out of our scope.

Another branch of methods is more similar to the approach described in this work and founded on class-based language models as introduced by Brown et al. [4]. The original class-based model groups n-grams of similar contexts in order to replace the probability of a certain word transition with the product of a class transition and the word emission probability:

$$P(w_n|w_1 \dots w_{n-1}) = P(c_n|c_1 \dots c_{n-1}) \cdot P(w_n|c_n). \quad (2.1)$$

The emission probability $P(w|c)$ can be estimated from a training corpus by dividing the number of tokens of word w by the number of tokens of all words of class c . This estimation corresponds to a maximum likelihood approach. Maltese et al. [15] use a supervised tagger and lemmatizer to define three back-off trigram models on words, part of speech tags (POS) and lemmas. They argue that the tag and lemma models only outperform the word model

in cases of bad statistics. For this reason, they group the interpolation parameters depending on the order of the n-gram that is used by the word model to estimate the specific sequence. They expect a dominance of the word model for trigrams and a higher contribution of the tag and lemma models, if the word model has to back-off to bi- or even unigrams. The parameters are estimated using the expectation maximization algorithm. In the performed experiments on Italian corpora of different domains, the achieved improvement in perplexity² on a pure word model drops with a rising number of tokens from -8.8% (1 million tokens) to - 5.6% (107 million tokens).

Crespo et al. [6] use a similar approach to build the language model for a Spanish speech recognition system. The language model interpolates between a word model and a pseudo word class model, defined over a knowledge-engineered automatic tagger. They claim to reduce the word error rate (WER) on a Spanish speech recognition task by 26%.

Uebler et al. [27] use prefixes and suffixes to define morphological word classes. They apply a morphological decomposition algorithm (MALAGA) that is enhanced by a number of thresholds to segment words into prefix, stem and suffix. In a speech recognition task of German and Italian phrases an improvement of WER by 60% is achieved, compared to a word based bigram approach. However, they do not compare their results with a state-of-the-art language model and omit a discussion of the handling of out-of-vocabulary words and n-gram discounting.

²Perplexity is a standard measure of language model performance. Consult section 4.1.5 for a definition.

Chapter 3

Feature Design and Clustering

In this chapter we define a set of features to map words to real number vectors, which then can be clustered by a general clustering algorithm. In this work, the k-means algorithm is used. Different combinations of feature vectors and their corresponding cluster solutions are then evaluated against a reference classification. As there is no single accepted cluster evaluation measure - like for example perplexity in language modeling - we use a couple of criteria, which are also defined and discussed in this chapter. The goal of the features is to group words of a vocabulary into a partition similar to their part-of-speech (POS) classes. In this work, the Penn treebank tagset is used. A short description can be found in appendix B. Our features are divided into morphological and shape-like features.

3.1 Morphological Features

Our morphological features are based on flat morphological word decompositions. These decompositions are learned by an unsupervised general purpose algorithm called Reports.

3.1.1 The Reports Algorithm

The Reports algorithm [13] learns affixes from an unannotated corpus and segments words into a form suitable for the extraction of affix- and stem-based features 3.1.2. It was designed to be simple and almost parameter-less. While it works decently for languages with a simple morphology (particularly English), modifications are necessary to make it suitable for more complicated and productive languages like German, Finnish and Turkish. Such modifications were proposed by Demberg [8]. The basic algorithm works as

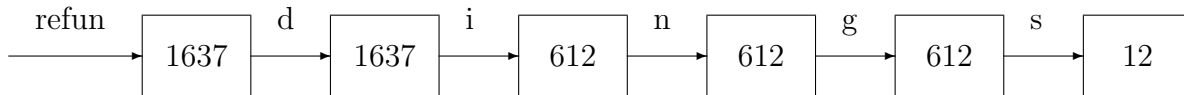


Figure 3.1: Example of the transitional probabilities in the word refundings indicating “-ing” and “-s” as good suffix candidates.

follows. A list of suffix candidates is estimated using the probability that a certain character succeeds a certain string. Low transitional probabilities are good indicators for morpheme borders. A suffix $s = xu$ is a candidate, if in at least 5% of its occurrences as the last part of a word $w = vyxu$:

- $P(x|vy) < 1$
- $P(y|v) \sim 1$
- The frequency of vy in the corpus is higher than a specific threshold

As shown in figure 3.1, “-ing” is a good suffix regarding the word “refundings”, as the probability $P(i|refund)$ is lesser than one, the probability $P(d|refun)$ is one and “refund” occurs quite often in the corpus. In a pruning step all suffixes which are concatenations of two other suffix candidates are removed from the list. In this work, we further reduce the list, to the 100 most frequent candidates. Prefixes are treated in an analog way. The strongest assumption the Report algorithm makes and the reason it performs purely on morphological complex languages is the third condition from the list above, i.e. that the word stem is a regular word of the target language.

The segmentation uses a rather rough heuristic: of all possible suffixes of a word, the one minimizing the probability $P(x|wy)$ is chosen. There is no mechanism to inhibit the clipping of a suffix. So if the word at hand was “for” and “or” was in the list of suffixes, then the segmentation would be $f + or$ ¹. To reduce the number of conflicts between prefixes and suffixes, in

¹Unless the suffix “r” was in the list and had a lower transition probability

the segmentation of a word of the length n , suffixes can only be cut from the $n/2 + 1$ back most characters of the word and prefixes accordingly from the $n/2 + 1$ foremost characters. This convention allows “being” to be segmented into “be+ing”, but also yields segmentations like “sur+f+ace”, where “ace” is not even in the suffix list, but “-face” is.

The Reports algorithm uses a weak definition of affix, e.g., a prefix is defined as a morpheme which occurs at the beginning of a number of words. “work”, for example, most probably appeared in the prefix and the suffix list (consider “workbench” and “homework”) and both cases would be correct, regarding this softer definition of an affix.

For the definition of four affixation-based features, we need to segment every word into prefix, stem and suffix, where prefix and suffix are allowed to be the empty word ϵ , but the stem is not. From the discussion of the Reports algorithm in the last chapter arise some issues we solve in a clean and simple way. The two problematic cases are inconsistent segmentations like “sur+f+ace” and segmentations with an empty stem like “home+work”. In both cases we prefer the suffix over the prefix and define the none-suffix part of the word as the stem. We do so because we regard the suffix as the most important part of the segmentation.

3.1.2 Affixation-based Features

Having a robust and unsupervised segmentation algorithm, we define two independent feature groups *Prefix* (P_1) and *Suffix* (S_1) over the abstract feature group *Affix* as an integer vector of the length 101, with:

$$\begin{aligned} \text{index}(w) &:= \begin{cases} 0, & \text{if } \text{affix}(w) = \epsilon \\ \text{index}(\text{affixes}, \text{affix}(w)), & \text{else} \end{cases} \\ \text{Affix}(w)[i] &:= \begin{cases} 1, & \text{if } i = \text{index} \\ 0, & \text{else} \end{cases} \end{aligned} \quad (3.1)$$

With affixes as the list of the 100 most frequent affixes and $\text{index}(l, x)$ as the index of x in the list l .

Thus, the feature is a mapping of a nominal attribute to a Boolean vector. As an example: given the suffix list: ϵ , “-s”, “-ing”, “-ed”, \dots , we would get:

$$\begin{aligned} \text{Suffix}(\text{“book”}) &= (1, 0, 0, 0, \dots) \\ \text{Suffix}(\text{“booking”}) &= \text{Suffix}(\text{“rebooking”}) \\ &= (0, 0, 1, 0, \dots) \end{aligned}$$

From a linguistic-morphological point of view, the prefix of an English word gives little information about its word type. Nonetheless, we add the Prefix feature to the feature vector to check if there are any useful statistical correlations. The last suffix of a word, however, determines the type of a large group of words in English and other languages, so we expect it to be rather important.

3.1.3 Stem-based Features

To separate ambiguous affixes, like “-s” or “-ing”, we introduce another feature to estimate which word type the stem of a word belongs to. Again we define two features $Prefix_Stem(P_2)$ and $Suffix_Stem(S_2)$ with an abstract feature $Affix_Stem$.

$$Affix_Stem(w)[0] := |\{w' \mid stem(w') = stem(w) \wedge affix(w') = \epsilon\}| \quad (3.2)$$

$$Affix_Stem(w)[i > 0] := |\{w' \mid stem(w') = stem(w) \wedge affix(w') = affixes[i]\}| \quad (3.3)$$

Thus, the affix stem feature is a vector, where every component corresponds to a specific affix and denotes how many word types are decomposed into the particular stem and affix. The first component belongs to the empty affix. Thus, given the suffix list from above: ϵ , “-s”, “-ing”, “-ed”, \dots and with only “book”, “book+s”, “book+ing” and “re+book+ing” in the vocabulary we get

$$\begin{aligned} Suffix_Stem(\text{“book”}) &= Suffix_Stem(\text{“booking”}) \\ &= (1, 1, 2, 0, \dots) \end{aligned}$$

3.2 Shape-like Features

Shape-like features can be found in natural language processing applications such as named entity recognition. Most of the features are designed to identify proper nouns and cardinal numbers.

3.2.1 Capitalization

The Capital Group and Special Group are vectors of Boolean predicates. They are transformed into real vectors by mapping the truth values *false*

and *true* to 0 and 1, respectively.

$$\begin{aligned}
\text{is_capital}(w) &:= \text{first character of } w \text{ is an uppercase letter} \\
\text{is_all_capital}(w) &:= \forall c \in w : c \text{ is an uppercase letter} \\
\text{capital_character}(w) &:= \exists c \in w : c \text{ is an uppercase letter} \\
\text{appears_in_lowercase}(w) &:= \neg \text{capital_character}(w) \vee w' \in \Sigma_T
\end{aligned}$$

Where w' is obtained from w by changing all uppercase letters to their lowercase counterparts and Σ_T is the vocabulary of a training corpus. Thus, with “book”, “Book”, “Rebook” and “REBOOK” but not “rebook” in the vocabulary we would get:

$$\begin{aligned}
\text{Capital_Group}(\text{“book”}) &= (0, 0, 0, 1) \\
\text{Capital_Group}(\text{“Book”}) &= (1, 0, 0, 1) \\
\text{Capital_Group}(\text{“REBOOK”}) &= (1, 1, 1, 0)
\end{aligned}$$

3.2.2 Special Characters

Along the lines of the Capital Group we define the Special Group using three further predicates:

$$\begin{aligned}
\text{special_character}(w) &:= \exists c \in w : c \text{ is a special character} \\
\text{digit}(w) &:= \exists c \in w : c \text{ is a digit} \\
\text{is_number}(w) &:= w \in L([+ - \epsilon][0 - 9] (([.,][0 - 9])[0 - 9]) *)
\end{aligned}$$

With $L[expr]$ as the language generated by the regular expression *expr*. As the resulting vectors have to be normalizable, i.e. different from the null vector, we add the predicate *not_special*, defined as the negated conjunction of the three original special group predicates, to the special group. Note, that nothing alike is necessary for the Capital Group, as a word violating the first three predicates must fulfill the last one. While many languages use the Hindu-Arabic numerals to represent a majority of the numbers in the written language, the almost exclusive usage of capitalization to mark proper nouns is seen more rarely, on that account, the Capital Group features may not work for languages not having this property. Independently of the vocabulary we get:

Special_Group("book")	=	(0, 0, 0, 1)
Special_Group("AT&T")	=	(1, 0, 0, 0)
Special_Group("C\$1.5")	=	(1, 1, 0, 0)
Special_Group("100,000")	=	(1, 1, 1, 0)
Special_Group("42")	=	(0, 1, 1, 0)

3.2.3 Word Length

As mentioned above, an important part of the word types cannot be clustered satisfactorily, as the words of these groups do not share sufficient morphological information. Representative of these types are articles, pronouns, interjections, verbs and a some nouns. Using the fact that their average length differs, we try to reduce the similarity between the smaller word types, like pronouns and larger ones like nouns and verbs. That is, we use the fact that very small words have a special word distribution. With n as the length of the word w , the feature Length is defined as:

$$\text{Length}(w)[i] := \min(n, 4) = i \quad (3.4)$$

For example:

Length("a")	=	(1, 0, 0, 0)
Length("be")	=	(0, 1, 0, 0)
Length("sea")	=	(0, 0, 1, 0)
Length("deal")	=	(0, 0, 0, 1)
Length("eagle")	=	(0, 0, 0, 1)

3.3 K-Means Clustering

We cluster using the bisecting k-means algorithm, a variation of k-means. The k-means algorithm estimates for a sequence of data points $D = \{\vec{x}_i\}$ a vector of centroids $\vec{v} = (\vec{v}_1, \dots, \vec{v}_k)$, approximately minimizing the overall distance between every data point and its closest centroid, which is the vector \vec{v} minimizing the function:

$$f[D](\vec{v}) = \sum_i \min\{(\vec{v}_j - \vec{x}_i)^2 | 1 \leq j \leq k\}. \quad (3.5)$$

The standard form of the algorithm was stated in [14] by Stuart Lloyd and mainly consists of alternately calculating centroids and clustering from each other:

1. Start with an initial vector \vec{v}' (e.g. k random elements of \vec{x}_i)
2. Calculate $\vec{c} = (c_1, \dots, c_k)$, where c_i is the set of points closest to \vec{v}_i
3. Calculate \vec{v} as the centroids of \vec{c}
4. Repeat 2. and 3. until a stopping criterion is reached

Lloyd’s algorithm only estimates a local minimum of f and depends heavily on the initial centroids v' . Steinbach et al. [26] introduced a procedure to find effective initialization vectors, called bisecting k-means. Starting with one cluster holding all data points, the initial vector is estimated by repeating the following scheme k times:

1. Select a cluster to split
2. Find 2 sub-clusters using the basic k-means algorithm.
3. Repeat 2. a couple of times and take the split with the highest overall similarity.

Steinbach et al. propose a number of selecting criteria including the largest cluster or the one with the least overall similarity. In this work, the largest cluster is split.

The bisecting k-means algorithm can be used to generate hierarchical and flat clusterings. As an example we look at a hierarchical clustering that was calculated using the features Capital_Group, Special_Group, Length and Suffix. The resulting clustering can be seen in figure 3.2, 3.3 and 3.4. Every leave of the complete tree in 3.2 corresponds to one cluster of the flat clustering we use in the evaluation and in the language model. For the sake of simplicity every flat cluster is represented by one of its words.

From the figure we can derive a precedence of our features. The first branching isolates words without special characters that also occur uncapitalized (cf. figure 3.2 and 3.3). The next couple of splits (with some exceptions) extract the words matching the is_number format (cf. figure 3.4).

In the later evaluation and the entire language model we only use flat clusterings.

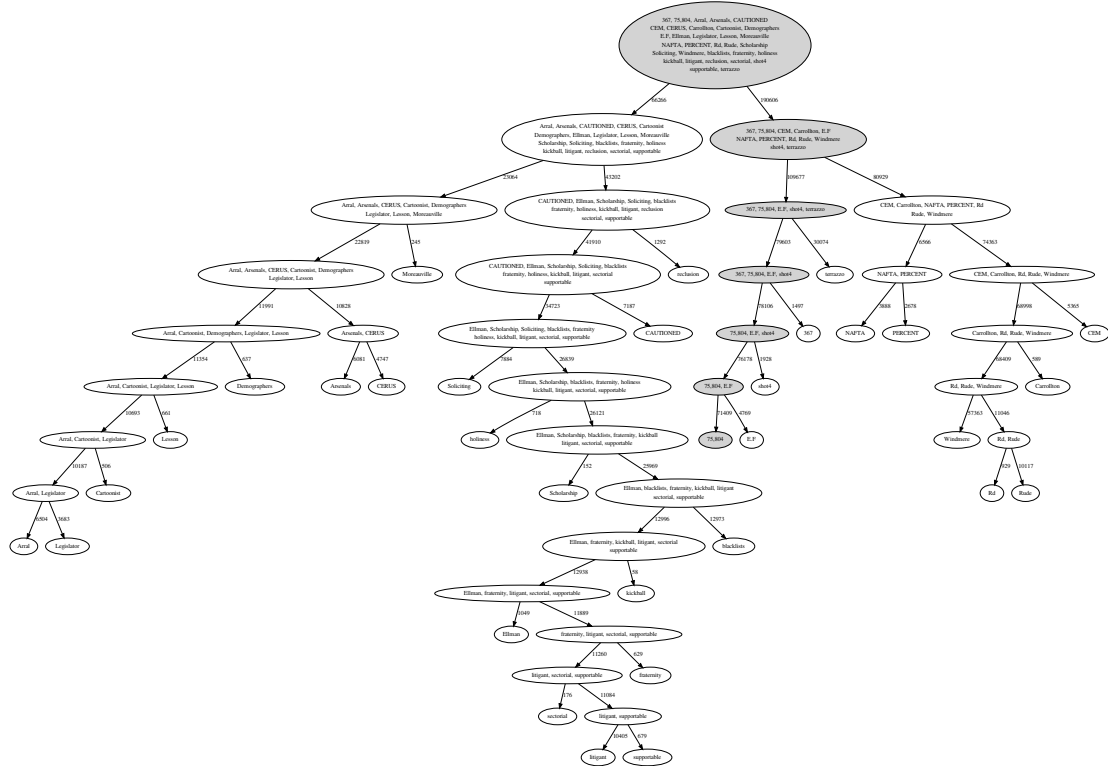


Figure 3.2: An hierarchical clustering of the training set of section 4. The edge weights correspond to the number of word types of the cluster the edge is pointing to.

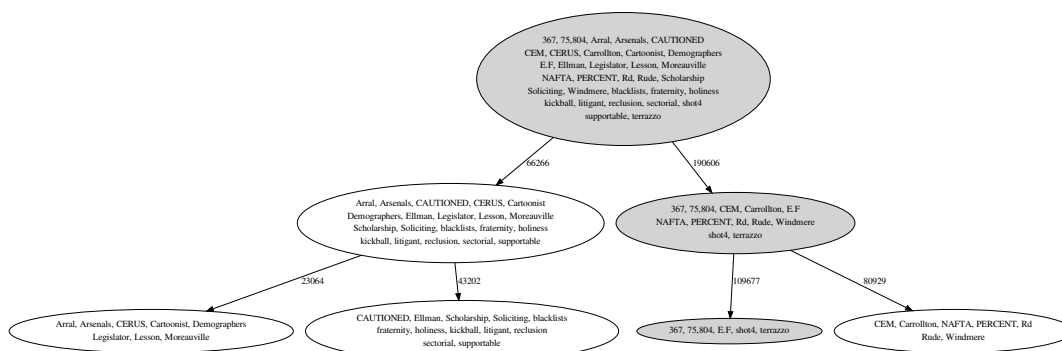


Figure 3.3: The three uppermost layers of the same clustering as in 3.2.

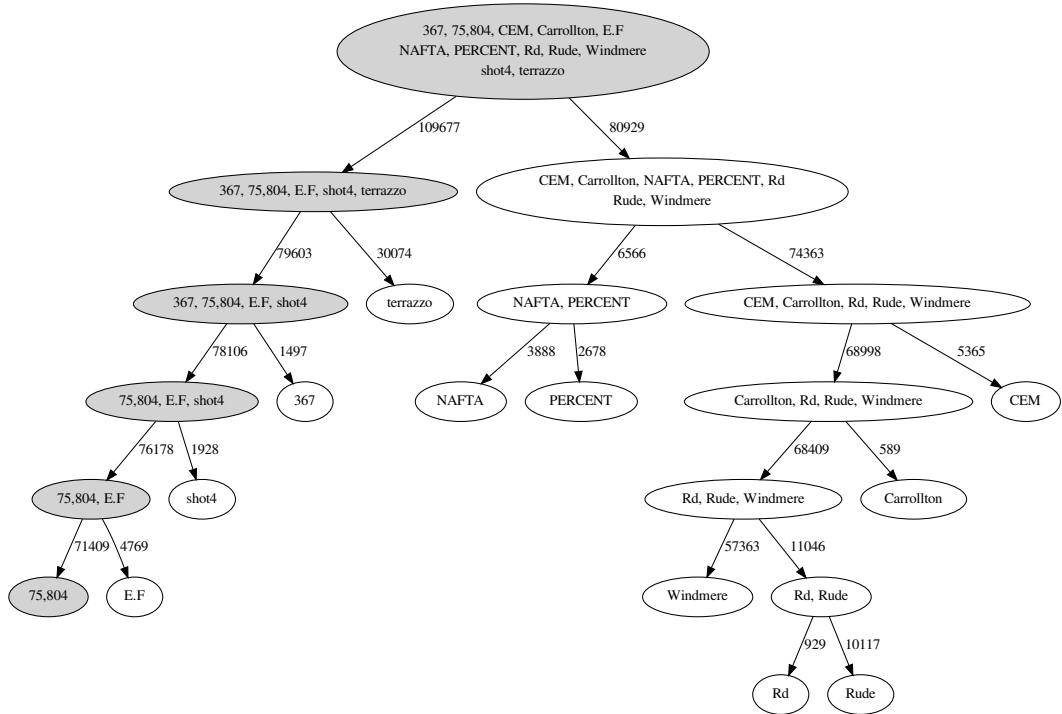


Figure 3.4: Subtree of the right uppermost child node of the graph shown in 3.2.

3.4 Evaluation Criteria

We evaluate clusterings by comparing them with a gold classification, obtained by grouping the words by their part-of-speech tags (POS). The tags are obtained by Treetagger [24]. Of course, the tag class of a word depends on its context. That is why we get a tag distribution for every word, which then has to be reduced to a single tag. We do this by taking the most frequent tag. However, for some tag classes, like past participle and past tense a small fluctuation might decide the tag class, as they are usually equally frequent.

In this work, several criteria were used as there exist certain issues with some criteria and others are not widely used. The quantities we use are the pair-based measures F-Measure and Adjusted Rand Index (ARI) and Q_2 -Measure, V-Measure and the Variation of Information (VI), which are motivated by information theory. Measures from information theory and pair-based mea-

sures are both calculated upon the contingency matrix, which states how many words of a cluster can be found in a certain class and vice versa. For a partitional clustering $K = \{k_i\}$ and a set of classes $C = \{c_i\}$ the two-dimensional contingency matrix $A = \{a_{ij}\}$ is defined as:

$$a_{ij} = |c_i \cap k_j| \quad (3.6)$$

3.4.1 Pair-based Criteria

Pair-based measures count the pairs of data points on which clustering and classification agree or disagree. They are defined over the reduced contingency matrix $A' = \{a'_{ij}\}$, which is a 2×2 matrix, where a'_{00} is the number of pairs of vectors that are in the same class and in the same cluster. a'_{01} is the number of pairs in the same class, but not in the same cluster, a'_{10} accordingly the number of pairs in the same cluster, but not in the same class and a'_{11} is the number of pairs not in the same class and not in the same cluster. Considering a data set of n vectors, the total number of pairs is given by $\binom{n}{2}$. The elements of the reduced contingency matrix can then be formulated as follows²:

$$a'_{00} = \sum_i \sum_j \binom{a_{ij}}{2} \quad (3.7)$$

$$a'_{01} = \sum_i \binom{a_{i\bullet}}{2} - a'_{00} \quad (3.8)$$

$$a'_{10} = \sum_j \binom{a_{\bullet j}}{2} - a'_{00} \quad (3.9)$$

$$a'_{11} = \binom{n}{2} - a'_{00} - a'_{01} - a'_{10} \quad (3.10)$$

Where $a'_{i\bullet}$ denotes a summation over \bullet , e.g. $a_{i\bullet} = \sum_j a_{ij}$. Apparently, $a'_{00} + a'_{11}$ is the number of correct pair decisions and $a'_{01} + a'_{10}$ the number of wrong pair decisions. Precision and Recall can then be defined as:

$$P = \frac{a'_{00}}{a'_{00} + a'_{10}} \quad (3.11)$$

$$R = \frac{a'_{00}}{a'_{00} + a'_{01}} \quad (3.12)$$

²The notation is inspired by [9]

Neither Precision nor Recall are good criteria on their own, because they can be optimized trivially. Precision is optimized by a clustering that puts every data vector in its own cluster and Recall is optimized by putting all vectors into the same cluster. Usually, one combines the two aspects to receive a more comprehensive measure. The F-measure for example is the weighted harmonic mean of the two.

$$F_\beta = \frac{(1 + \beta) \cdot P \cdot R}{\beta \cdot P + R} \quad (3.13)$$

Another well known measure is the Rand index (RI) [20]. It is defined as the number of right decision divided by the total number of decisions, i.e. the number of pairs.

$$RI = \frac{a'_{00} + a'_{11}}{\binom{n}{2}} \quad (3.14)$$

Awkwardly, the clusterings achieved in practice lie in a small interval around 0.9. In [11] Hubert and Arabie proposed a renormalization to a random clustering as base-line, instead of a clustering without right decisions. This renormalization is done by the general form for an index “corrected for chance”:

$$\frac{Index - E[Index]}{Max[Index] - E[Index]}. \quad (3.15)$$

The adjusted index is bounded by 1, is 0 if the index equals its expectation value and is negative if the index is lesser than its expectation value. However, as mentioned above, in a normal experimental setting, the adjusted index will lie above 0. It was shown in [11] that the expectation value of the RI in a generalized *hyper-geometric* distribution is

$$1 + 2(a'_{00} + a'_{10})(a'_{00} + a'_{01}) / \binom{n}{2} - (2a'_{00} + a'_{01} + a'_{10}) / \binom{n}{2}. \quad (3.16)$$

With eq. 3.16 and 1 as the maximum of the Rand index, the adjusted Rand index can be written as:

$$\begin{aligned} ARI &= \frac{a'_{00} - (a'_{00} + a'_{10})(a'_{00} + a'_{01}) / \binom{n}{2}}{\frac{1}{2}(2a'_{00} + a'_{01} + a'_{10}) - (a'_{00} + a'_{10})(a'_{00} + a'_{01}) / \binom{n}{2}} \\ &= \frac{(a'_{00} \cdot a'_{11} - a'_{01} \cdot a'_{10})}{(a'_{00} + a'_{10}) \cdot (a'_{10} + a'_{11}) + (a'_{00} + a'_{01}) \cdot (a'_{01} + a'_{11})}. \end{aligned} \quad (3.17)$$

During the development of criteria based on information theory, some issues with pair-based measures were found. Dom argues in [9] that a usual clustering can be divided into clusters, correlated and uncorrelated with the classification. These two types are called useful and noisy. He demands of an evaluation criterion that it worsens with a rising number of uncorrelated clusters. He further shows empirically that the adjusted Rand index falls short on this meta criterion. Meila [17] shows that the ARI is not local, i.e. the change caused by splitting a cluster, does not only depend on the particular cluster. However, Meila acknowledges whether locality is a good property or not, depends heavily on the specific application.

3.4.2 Information-Theory-based Criteria

The criteria derived from information theory are usually based upon the conditional entropy $H(C|K)$, i.e. given the Clustering K how much information bits per data point are missing to determine the classification C exactly. In information theory, the entropy of a random variable C is defined as the expectation value of its information $-\log_2 P(C)$:

$$H(C) = - \sum_{c \in \text{val}(C)} P(c) \cdot \log_2 P(c). \quad (3.18)$$

Where $\text{val}(C)$ denotes the domain of C . From this definition, we can now derive the formulas for the joined and conditional entropies of two random variables C and K .

$$\begin{aligned} H(C, K) &= - \sum_{c \in \text{val}(C)} \sum_{k \in \text{val}(K)} P(c, k) \cdot \log_2 P(c, k) \\ H(C|K) &= - \sum_{c \in \text{val}(C)} \sum_{k \in \text{val}(K)} P(c, k) \cdot \log_2 P(c|k) \\ &= H(C, K) - H(K) \end{aligned} \quad (3.19)$$

The probabilities can be estimated with a maximum likelihood approach and using the complete contingency table from above. With n as the number of data points in the clustering ($n = a_{\bullet\bullet}$) the guessed probabilities are:

$$\begin{aligned}
\hat{P}(c_i) &= a_{i\bullet}/n \\
\hat{P}(k_j) &= a_{\bullet j}/n \\
\hat{P}(c_i, k_j) &= a_{ij}/n \\
\hat{P}(c_i|k_j) &= a_{ij}/a_{\bullet j} \\
\hat{P}(k_j|c_i) &= a_{ij}/a_{i\bullet}
\end{aligned}$$

And we finally get

$$\hat{H}(C) = - \sum_{c_i \in C} \frac{a_{i\bullet}}{n} \cdot \log_2 \frac{a_{i\bullet}}{n} \quad (3.20)$$

$$\hat{H}(K) = - \sum_{k_j \in K} \frac{a_{\bullet j}}{n} \cdot \log_2 \frac{a_{\bullet j}}{n} \quad (3.21)$$

$$\hat{H}(C|K) = - \sum_{c_i \in C, k_j \in K} \frac{a_{ij}}{n} \cdot \log_2 \frac{a_{ij}}{a_{\bullet j}} \quad (3.22)$$

$$\hat{H}(K|C) = - \sum_{c_i \in C, k_j \in K} \frac{a_{ij}}{n} \cdot \log_2 \frac{a_{ij}}{a_{i\bullet}} \quad (3.23)$$

for the estimated entropies. As with Precision and Recall of the pair-based measures, the conditional entropy $H(C|K)$ is a poor criterion on its own. Like Precision it can be optimized (i.e. minimized) by putting every data point in its own cluster that all probabilities $P(c|k)$ are either 0 or 1, and thus, all the terms $P(c, k) \cdot \log_2 P(c|k)$ result in 0. However, $H(C|K)$ is a fine measure if the number of clusters to partition the data points in is fixed. A first approach for a criterion that could compare clusterings of different sizes, was proposed by Dom in [9], where he defines an additional adjustment term, to rank clusterings of the same conditional entropy $H(C|K)$ but different cluster counts.

$$Q_0(C, K) = \hat{H}(C|K) + \frac{1}{n} \sum_{k_j \in K} \log_2 \binom{a_{\bullet j} + |C| - 1}{|C| - 1}. \quad (3.24)$$

Dom argues that $H(C|K)$ bits per object are insufficient to calculate C from K , because a decoder needs to know the exact relation between C and K also. This relation is stored in the contingency table A . However, the decoder does not need the entire table, as he already knows its row sums. The missing information of the j th row of A is the number of possible vectors of length $|C|$ summing to $a_{\bullet j}$, i.e. the size of cluster k_j . It is given by the combinatorial

term

$$\binom{a_{\bullet j} + |C| - 1}{|C| - 1}. \quad (3.25)$$

The bits necessary to transmit the entire contingency table, per data point, are then given by the second term in eq. 3.24. Recent works [21,22] argue that a comprehensive measure should combine two aspects called homogeneity and completeness, as equally important components. Homogeneity means that all objects in one cluster have the same class. Analogously, completeness means that all objects of the same class are assigned to a single cluster³. The Q_0 measure represents both ideas. However, it permits to trade off a lower value in the completeness term (the second term), for a higher value in the homogeneity term (the first term), as homogeneity and completeness contribute unequally to the result. Dom also defined Q_2 , a modification of Q_0 , which is normalized to the interval $[0, 1]$ with $Q_2(C, K) = 1$, if and only if C equals K . Q_2 is easier to use together with other measures like F measure, V measure and ARI, therefore we use it to evaluate the cluster experiments in this work.

$$Q_2(C, K) = \frac{\frac{1}{n} \sum_{c_i \in C} \log \binom{a_{i\bullet} + |C| - 1}{|C| - 1}}{Q_0(C, K)} \quad (3.26)$$

Latest papers introduce the conditional entropy $H(K|C)$ to estimate completeness. Meila, for example, introduced the variation of information (VI) [17]:

$$VI(C, K) = H(C|K) + H(K|C) \quad (3.27)$$

and showed that it is a metric on the space of clusterings. As VI is a metric its lower bound is 0, achieved if and only if the clustering equals the classification. Meila shows that the upper bound of VI is $\log n$ and for an upper bound of the number of clusters K^* , there exists a smaller upper bound $2 \log K^*$. However, it is inappropriate to divide VI by $\log n$ to create a normalized version of VI, as it would be incomparable between data sets of different sizes. And of course there is a similar problem with normalizing by $2 \log K^*$. As mentioned in [21], the normalization by $\log n$ is particularly questionable, since a data set is not necessarily more complicated to cluster, if it holds more data. Consider that splitting for given C and K every data point into two would not change the VI, but its normalization would. Reichart and

³These definitions express similar ideas as Precision and Recall

$$\begin{aligned}
C &= \{\{o_i | 0 < i < 10\}, \{o_{10}\}\} \\
K &= \{\{o_i | 0 < i < 9\}, \{o_9, o_{10}\}\} \\
H(C) &= -\frac{9}{10} \cdot \log_2 \frac{9}{10} - \frac{1}{10} \cdot \log_2 \frac{1}{10} \approx 0.47 \\
H(K) &= -\frac{8}{10} \cdot \log_2 \frac{8}{10} - \frac{2}{10} \cdot \log_2 \frac{2}{10} \approx 0.72 \\
H(C, K) &= -\frac{8}{10} \cdot \log_2 \frac{8}{10} - \frac{2}{10} \cdot \log_2 \frac{1}{10} \approx 0.92 \\
VI(C, K) &= H(C|K) + H(K|C) = 2H(C, K) - H(C) - H(K) \approx 0.65 \\
NVI(C, K) &= \frac{VI(C, K)}{H(C)} \approx 1.39
\end{aligned}$$

Figure 3.5: An example demonstrating that the NVI can be greater than 1 for decent clusterings.

Rappoport [21] propose a normalized version of the VI, defined as:

$$NVI(C, K) = \begin{cases} \frac{VI(C, K)}{H(C)}, & \text{if } H(C) \neq 0 \\ H(K), & \text{else} \end{cases} \quad (3.28)$$

There exist some issues with NVI in comparison with VI and other measures. Obviously, NVI is no longer a metric, as it is antisymmetric. Furthermore, NVI is unbounded, as in the no knowledge case, $H(C|K)$ equals $H(C)$ and thus NVI equals $1 + H(K)/H(C)$. Reichart and Rappoport argue that it is only greater than 1 for very bad clusterings, i.e. the single cluster solution. However, for a binary classification with low entropy, e.g. when one class is bigger than the other one, the NVI will usually be greater than 1, including for very good clusterings, where only one data point is misplaced. See figure 3.5 for an example. Moreover, we find NVI exceeding 1 for clusterings, other measures like V, F, ARI and Q_2 rate as proper. NVI measures the distance of clustering and classification against the uncertainty or complexity of the classification. However, it is hard to describe, on a theoretical basis, what NVI measures in terms of information theory, i.e. what the receiver-transmitter scenario would be.

A last information-theory-based measure is V-measure [22]. V-measure, like F-measure, is the weighted harmonic mean of the explicitly defined quantities homogeneity (h) and completeness (c). h and c are defined over the

normalized conditional entropies:

$$h = \begin{cases} 1, & \text{if } H(C, K) = 0 \\ 1 - \frac{H(C|K)}{H(C)}, & \text{else} \end{cases}$$

$$c = \begin{cases} 1, & \text{if } H(C, K) = 0 \\ 1 - \frac{H(K|C)}{H(K)}, & \text{else} \end{cases}$$

The joined probability $H(C, K)$ is zero, if and only if classification and clustering are equal.

$$V_\beta = \frac{(1 + \beta) \cdot h \cdot c}{\beta \cdot h + c} \quad (3.29)$$

Even though, one could argue that homogeneity is more important for our application than completeness, all the evaluation here is done with the measure V_1 . The quantity V_1 equals the normalized mutual information or *symmetric uncertainty* as defined in [29]. A proof can be found in the appendix A.

$$I(C, K) = H(C) + H(K) - H(C, K)$$

$$\text{NI}(C, K) = \frac{2 \cdot I(C, K)}{H(C) + H(K)}$$

3.4.3 Limitations of Contingency-Table-based Criteria

All the indexes we discussed here weight every single data point equally important. Although being a good general assumption, this does not fit well with the task at hand. The problem is that the performance of a language model depends on the precise estimation of clusters with a high token frequency, while the criteria prefer clusters with a high type count. Of course, one could just weight the word types by their token frequencies. However, this would result in a clustering that - as Zipf’s law for natural languages states - is dominated by high frequent tokens, which are mostly of types that can’t be clustered morphologically at all. Consider the following example: for a part of an English corpus⁴ of 39.027.486 tokens the ten most frequent types sum up to a total token frequency of 10.916.672 ($\approx 28\%$). These word types are: “.”, “,”, “the”, “of”, “to”, “-”, “a”, “and”, “in”, and the single quote. They are only affected by two of our features (the special character feature and the length feature) and any grouping of them will lead to an inferior performance of the language model. So applying frequency weights to the evaluation would have a somehow random affect, which would not scale to better language model performance.

⁴The corpus is a part of the Wall Street Journal corpus, which we also use to train the languages models of chapter 4.

3.5 Experimental Setup and Results

In this section, we compare different combinations of the features defined at the beginning of this chapter. We use Treetagger to generate a gold standard classification, which is then compared with clusterings produced by the bisecting k-means algorithm. We only expect our features to work for a small part of the 56 tag classes assigned by Treetagger. These classes include cardinal numbers, proper nouns, plural nouns, regular verb forms and a subset of classes with a more heterogeneous morphology like adjectives, verb base forms, non-third person present forms and nouns. The number of clusters k is 56 for all experiments. We do not compare all possible feature combinations. The procedure we apply here corresponds more to a greedy algorithm than to a deep analysis. We start with a core vector of features we assume to be irreplaceable. These features are the Capital Group (CG), the Special Group (SG) and the suffix group (S_1). As explained before, CG and SG are the only features we have to identify the important groups of proper nouns and cardinal numbers. Furthermore, English morphology tells us that suffixes are most important for the estimation of a word’s class. Starting with the initial core vector, we pick an additional feature, we test if adding the feature to the core vector improves the clustering results and if so, we keep it in the feature vector. As said in the section above, the huge number of cardinal numbers and proper nouns (about 65% of all word types) has a strong impact on the evaluation criteria reducing the influence of the rest of the features. This only leads to small effects on the absolute cluster rating. The first feature we test is the Length feature L . The comparison of the feature vectors with and without L can be seen in table 3.1.

Feature vector	F1	ARI	Q2	V	VI
(C, Sp, S_1)	0.6574	0.5634	0.0230	0.4755	2.9222
(C, Sp, S_1, L)	0.6642	0.5882	0.0332	0.5538	2.7761
Relative Difference	+1.03%	+4.4%	+44.35%	+16.47%	-5%

Table 3.1: Performance of the Length feature.

We can see that F-score and adjusted Rand index (ARI) have a more conservative opinion about L , but the IT measures Q_2 and V evidence a rather strong improvement of 44% and 17% respectively. It is common in all our results that the five criteria agree on whether a feature improves the clustering or not, but they disagree in the quantification of the improvement. As a consequence of the results we keep L in the vector and proceed with the next feature P_1 . The corresponding comparison can be seen in table 3.2.

Feature vector	F1	ARI	Q2	V	VI
(C, Sp, S_1, L)	0.6642	0.5882	0.0332	0.5538	2.7761
(C, Sp, S_1, L, P_1)	0.6513	0.5742	0.0305	0.5331	2.8733
Relative Difference	-1.94%	-2.38%	-8.13%	-3.74%	3.5%

Table 3.2: Performance of the feature P_1

All criteria imply a small negative effect of P_1 . So we remove it from the vector and proceed with the last features P_2 and S_2 . The results for P_2 and S_2 can be found in the same tables 3.3 and 3.4.

Feature vector	F1	ARI	Q2	V	VI
(C, Sp, S_1, L)	0.6642	0.5882	0.0332	0.5538	2.7761
(C, Sp, S_1, L, S_2)	0.6484	0.5720	0.0258	0.4553	3.5370
Relative Difference	-2.38%	-2.75%	-22.29%	-17.79 %	27.41%

Table 3.3: Performance of the stem feature S_2

Feature vector	F1	ARI	Q2	V	VI
(C, Sp, S_1, L)	0.6642	0.5882	0.0332	0.5538	2.7761
(C, Sp, S_1, L, P_2)	0.6437	0.5650	0.0257	0.4664	3.3597
Relative Difference	-3.09%	-3.94%	-22.59%	-15.78%	21.02%

Table 3.4: Performance of the stem feature P_2

All criteria agree that the stem features S_2 and P_2 produce inferior clusters. This can be explained by the fact that the Affix_Stem features reduce the distance between words of different classes, if they have the same stem or a stem which is used with the same affixes. For example, regarding the features S_2 the differences between the words *book* and *booking* is zero. As a result, the similarity between all verb form classes is increased. On the other hand the features reduce the similarity between pure verb stems like *sing*, stems that occur frequently in verbs and nouns like *work* and mainly noun stems like *word*. So by (partially) segmenting the classes VVZ and NNS, the classes VVZ, VVD, VVP and VV are blended, which leads to worse clustering results. One could say that the stem features represent a semantic concept and mixing this aspect with the syntactic ideas behind the other features (and the syntactic nature of the POS classes) is counterproductive at this point.

Finally, we stop with the feature vector (CG, SG, S_1, L) . We would like to point out that the sketched procedure can be easily transformed into an

algorithm. To find a good feature vector for another language, one could start with the core vector (CG, SG, S_1) (which should be a good starting point for most languages) and test a series of other more language specific features. As an example we suppose that the prefix P_1 would be a good extension of the feature vector for German. The only thing missing for this to be a valid algorithm is a voting procedure to decide ties between the rating criteria.

Chapter 4

A Morphological Language Model

In this chapter, we use the clusterings from chapter 3 to define a morphological language model. We then compare this model with a state-of-the-art Kneser-Ney model implementation and propose a linear interpolation of the two models yielding a lower perplexity on a test corpus than the Kneser-Ney model itself. Before we explain our model in 4.2.1, we explain the foundations of language modeling, namely the principles of word modeling with n-grams using the maximum likelihood estimate, Kneser-Ney models and class-based models as introduced by [4]. Readers already familiar with language modeling may as well skip the next section.

4.1 Foundations of Language Modeling

A statistical language model estimates the probability of a sentence or a sequence of words $w_i = w_1 \cdots w_n = w_1^n$. This is usually done by modeling w_i using a left-to-right model or the so called *chain rule*:

$$\begin{aligned} P(w_1 \cdots w_n) &= P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1w_2) \cdots P(w_n|w_1 \cdots w_{n-1}) \\ &= \prod_{i=1}^n P(w_i|w_1 \dots w_{i-1}) \\ &= \prod_{i=1}^n P(w_i|w_1^{i-1}) \end{aligned} \tag{4.1}$$

Hence, to calculate the probability of a sentence one moves from left to right and calculates the probability of the next token w_i given that the preceding tokens w_1^{i-1} . The conditional term w_1^{i-1} is also called *history*. For a given

vocabulary V and a sentence of length n the last conditional probability in that chain $P(w_n|w_1^{n-1})$ corresponds to a probability table with $|V|^n$ entries. To prevent data-sparseness issues the maximal length of a history is bounded by a quantity $N - 1$ and thus

$$P(w_1 \cdots w_n) \approx \prod_{i=1}^n P(w_i|w_{i-N+1}^{i-1}). \quad (4.2)$$

Throughout this work we use trigram models ($N = 3$), i.e. the maximal history length is 2. The remaining problem is to estimate the conditional probabilities from a given training set D .

4.1.1 The Maximum Likelihood Estimate

To measure the performance of a model Θ it is common to use the probability of D under *Theta*. The so called likelihood (LL) is:

$$LL(D|\Theta) = \prod_{s \in D} P_{\Theta}(s). \quad (4.3)$$

It can be shown that a model trained on D maximizes the LL of D , if the word probabilities are chosen in the following way:

$$\hat{P}_{ML}(w_i|w_{i-N+1}^{i-1}) := \frac{C[D](w_{i-N+1}^i)}{C[D](w_{i-N+1}^{i-1})} \quad (4.4)$$

Where $C[D](x)$ is the frequency of the event x in D . Equation 4.4 is called the Maximum Likelihood estimate (MLE) of D . The MLE model is the best model for D , but is it also the best model for a set D' with a similar but not the same distribution? If D' occurred a trigram w_1^3 that did not occur in D , the LL of D' given the MLE model of D would be zero. To generalize the MLE and also cover such unseen events, we have to redistribute some of the probability of the events of D to all other events. This approach is called discounting and in the next section we are going to present one of the best discounting methods: Kneser-Ney discounting.

4.1.2 Kneser-Ney Discounting

As a complete derivation and justification would go beyond the scope of this work, we would like to refer the reader to the explanations in [12] and the excellent analysis done in [5] for a deeper understanding of this matter. The notation we use here is taken from [5]. Kneser-Ney discounting derives from a

simpler form of smoothing called *absolute discounting*. Absolute discounting [18] is motivated by the observation that subtracting a small discount d from the frequency of every event results in both a small relative change of high frequent events and a big relative change of low frequent events. This goes well with the intuition that high counts are more trustworthy than low counts. Absolute discounting can then be defined as an interpolated model between N-grams of different orders:

$$\begin{aligned} \hat{P}_{abs}(w_i|w_{i-N+1}^{i-1}) &:= \frac{C(w_{i-N+1}^i) - d}{C(w_{i-N+1}^{i-1})} \\ &+ \frac{d}{C(w_{i-N+1}^{i-1})} N_{1+}(w_{i-N+1}^{i-1} \bullet) \cdot \hat{P}_{abs}(w_i|w_{i-N+2}^{i-1}) \end{aligned} \quad (4.5)$$

Where $N_{1+}(w_{i-N+1}^{i-1} \bullet)$ is the number of different continuations of w_{i-N+1}^{i-1} . The discount d is estimated as a ratio between the number of N-grams with a frequency of one and two (n_1 and n_2 respectively).

$$d = \frac{n_1}{n_1 + 2 \cdot n_2}$$

Hence, the interpolation is defined recursively and ends when the unigram probability $\hat{P}(w_i)$ is reached. For a sequence w_1^3 with $C(w_1^3) = C(w_2^3) = 0$ we then get:

$$\hat{P}_{abs}(w_3|w_1^2) = \frac{d^2}{C(w_1^2)C(w_2)} N_{1+}(w_1^2 \bullet) N_{1+}(w_2 \bullet) \cdot \hat{P}_{abs}(w_3)$$

That is, the probability of a continuation depends only on the unigram probability of w_3 ¹. But this is not a good estimation. Consider bigrams like “New York” or “San Francisco”. They might cause that the unigram count of “York” and “Francisco” are high, even though they only occur in a very specific context. Kneser-Ney discounting attempts to target this problem by replacing the term $\hat{P}_{abs}(w_i|w_{i-N+2}^{i-1})$ in equation 4.5:

$$\begin{aligned} \hat{P}_{KN}(w_i|w_{i-N+1}^{i-1}) &:= \frac{C(w_{i-N+1}^i) - d}{C(w_{i-N+1}^{i-1})} \\ &+ \frac{d}{C(w_{i-N+1}^{i-1})} \cdot \frac{N_{1+}(w_{i-N+1}^{i-1} \bullet) N_{1+}(\bullet w_{i-N+2}^i)}{N_{1+}(\bullet w_{i-N+2}^{i-1} \bullet)} \end{aligned} \quad (4.6)$$

¹As $\hat{P}_{abs}(w_3)$ is the only term depending on w_3 .

Where accordingly to $N_{1+}(w_{i-N+1}^{i-1} \bullet)$, $N_{1+}(\bullet w_{i-N+2}^i)$ is the number of different words w_{i-N+1} preceding w_{i-N+2}^i and $N_{1+}(\bullet w_{i-N+2}^{i-1} \bullet)$ is the number of different pairs w_{i-N+1}, w_i respectively preceding and continuing w_{i-N+2}^{i-1} . That is, if $w_2 w_3$ does not occur in the training set D , then the probability of w_3 continuing the sequence $w_1 w_2$ is proportional to the number of different words preceding w_3 in D , i.e. $N_{1+}(\bullet w_i)$. Chen and Goodman presented a version of Kneser-Ney, they call modified Kneser-Ney [5]. They argue that the optimal estimate of the discount d varies drastically for ngrams of the counts 1, 2 and greater or equal to 3 and thus introduce three discounts: d_1 , d_2 , d_{3+} respectively.

4.1.3 Class-based Models

Class-based models are a further way to improve the generalization capabilities of language models. The main idea is to group words of similar contexts into classes and to treat them equally. The classical model presented by Brown et al. in [4] replaces the word transmission probability with:

$$P(w_i | w_{i-N+1}^{i-1}) = P(c(w_i) | c(w_{i-N+1}^{i-1})) \cdot P(w_i | c(w_i)) \quad (4.7)$$

Where $c(w)$ is the class of word w and $P(c(w_i) | c(w_{i-N+1}^{i-1}))$ is known as class transmission probability and can be modeled by any modeling technique, e.g. MLE or KN. The word emission probability $P(w_i | c(w_i))$ is defined as:

$$P(w | c(w)) = \frac{C(w)}{\sum_{w \in c(w)} C(w)} \quad (4.8)$$

Which is the MLE. As above, $C(w)$ denotes the token count of w . The core of a class-based model is of course the assignment of words to classes. However, we are not going to discuss the algorithm used in the original paper as we are going to replace it by a new one. In the literature many approaches have been proposed, for example classes based on part-of-speech tags or lemmas [15]. However, Niesler et al. demonstrated [19] that direct clustering of the word contexts outperforms such approaches. While class-based models improve the generalization of low frequent events, they also reduce the precision of the prediction of high frequent events. Thus, one often combines class-based models with state of the word models, such as KN models, to get the best of both concepts.

4.1.4 Linear Interpolation

Linear interpolation is a simple way of combining statistical models. In absolute discounting (eq. 4.5) and the KN model (eq. 4.6) linear interpolation

is used to integrate n-gram models of different order. In this section, we introduce a general interpolation scheme and an efficient and precise way of estimating the optimal interpolation parameters for a set of held-out data. For the sake of simplicity we reduce the number of interpolated models to two, as we are only combining two models in this work. The linear interpolation of two models is then given by:

$$P(w|h) = (1 - \lambda(h)) \cdot P_1(w|h) + \lambda(h) \cdot P_2(w|h) \quad (4.9)$$

In this setup, one parameter $\lambda(h)$ is introduced for every word history h . One attempts to reduce the number of parameters to an appropriate trade-off between precision and over-fitting. As in section 4.1.1 the likelihood can be used to estimate the performance of the interpolated model. An application of equation 4.3 yields:

$$LL[D](\lambda) = \prod_{w_i \in D} (1 - \lambda) \cdot P_1(w_i|w_{i-N+1}^{i-1}) + \lambda \cdot P_2(w_i|w_{i-N+1}^{i-1}). \quad (4.10)$$

For reasons of feasibility we replace the summation over N-gram tokens by a summation over N-gram types and apply the logarithm.

$$ll[D](\lambda) = \sum_{w_1^N} C[D](w_1^N) \cdot \log[(1 - \lambda) \cdot P_1(w_N|w_1^{N-1}) + \lambda \cdot P_2(w_N|w_1^{N-1})] \quad (4.11)$$

In most of the literature a version of the EM-algorithm [23] is used to find good parameters. However, a deeper analysis of the problem at hand, as done by [1] yields a faster and optimal algorithm. It can be shown that this log-likelihood function of an interpolated model is a concave function, e.g. that $-ll$ is a convex function in λ . This implies that ll has one extremum which is an optimum. This optimum can be found using the derivative $\partial ll / \partial \lambda$ and a fast bisecting interval search.

$$\frac{\partial ll[D](\lambda)}{\partial \lambda} = \sum_{w_1^N} C[D](w_1^N) \cdot \left(\lambda + \frac{P_2(w_N|w_1^{N-1})}{P_1(w_N|w_1^{N-1}) - P_2(w_N|w_1^{N-1})} \right)^{-1} \quad (4.12)$$

The algorithm starts searching in the interval $[0, 1]$ as only values in this interval are valid interpolation parameters. However, the global optimum may lie outside of this interval. This is the case if one of the two models performs better on all of the samples. The optimal parameter has to be either 0 or 1 in this case. If the derivative at 0 is negative, the global optimum

lies on the left side of the interval and 0 is the local optimum. The border value 1 can be treated analogously. If the global optimum lies inside the interval, it can be found by calculating the derivative at the middle. If the derivative is positive (negative), the search is continued on the right (left) side. The search ends after a certain number of iterations or when a point with a derivative of 0 is found. Because the maximal distance to the optimum is cut in half in every step, it is assured that the distance to the optimum after n iterations is smaller than 2^{-n} . See figure 4.1 for an implementation of the entire algorithm.

```
def estimate_parameter(freqs,prob_list1,prob_list2,epsilon):

    D      = lambda lambda : partial(ll, freqs, prob_list1, prob_list2)(lambda)
    n      = math.ceil(- log_2(epsilon))

    if D(0.) <= 0.:
        return 0
    if D(1.) >= 0.:
        return 1

    l = 0.
    r = 1.

    loop n:
        m = ( l + r ) / 2.
        d = D(m)
        if d == 0.:
            break
        elif d > 0.:
            l = m
        else: # d < 0.
            r = m

    return m
```

Figure 4.1: Implementation of Bahl's algorithm. The parameter ϵ is the desired maximal distance to the true optimum.

4.1.5 Language Model Evaluation

A widely used evaluation criterion for languages models is perplexity. Perplexity is motivated by information theory and closely related to the likelihood and cross entropy. Cross entropy is defined as the number of bits necessary to identify an event (in our case the occurrence of a word token) if the coding is based on the probability distribution P_θ (the language model) instead of the true probability distribution P_D (the test corpus).

$$\begin{aligned} H(P_D, P_\theta) &= - \sum_w P_D(w) \cdot \log P_\theta(w) \\ &= - \sum_w \frac{C[D](w)}{\sum_{w'} C[D](w')} \cdot \log P_\theta(w) \\ &= - \frac{l[D](\theta)}{\sum_{w'} C[D](w')} \end{aligned}$$

Perplexity is then defined as n raised to the power of the cross entropy where n is the base of the used logarithm, e.g

$$Perp(D, \theta) = 2^{-\sum_w P_D(w) \cdot \log_2 P_\theta(w)} \quad (4.13)$$

Hence, the language model θ maximizing the likelihood of a corpus D will minimize the cross entropy $H(P_D, P_\theta)$ and the perplexity $P(D, \theta)$. The language model that approximates a corpus the best will yield the lowest perplexity on it. One uses perplexity instead of likelihood to evaluate language models, as it is independent of the corpus size.

4.2 Design of a Morphological Language Model

Equipped with the general tools and the terminology of language modeling of section 4.1 and the morphological features developed in section 3 we develop a preliminary morphological model.

4.2.1 A Preliminary Model

The morphological language model uses a cluster solution $K = \{k_i\}$ to map every word to the index of its cluster: $c(w) = i \iff w \in k_i$ and the general form of a class-based model as introduced in equation 4.7:

$$P(w_i | w_{i-N+1}^{i-1}) = P(c(w_i) | c(w_{i-N+1}^{i-1})) \cdot P(w_i | c(w_i)) \quad (4.14)$$

To define the conditional probability $P_{morph}(c(w_i)|c(w_{i-N+1}^{i-1}))$ and the word emission probability $P_{morph}(w|c(w))$ two issues have to be taken care of: unknown words and unknown class n-grams. The problem with unknown words is that their word emission probability is zero and that the function c cannot assign a cluster label to them as they do not occur in the clustering. The latter can be fixed easily by calculating the feature vector of the unknown words w and choosing the cluster whose centroid is nearest to that vector. To solve the first problem we discount the emission probability of the known words and distribute it equally over the unknown words. To do so we estimate the fraction of unknown words ϵ using a validation corpus. Note that ϵ generally depends on the cluster and we use $\epsilon(c) = \epsilon$ for simplification. The emission probability for an arbitrary word is then defined as an extension of equation 4.8:

$$p(w_3|c_3) = \begin{cases} (1 - \epsilon) \frac{c(w_3)}{\sum_{w \in c_3} c(w)} & \text{if } c(w_3) > 0 \\ \epsilon & \text{if } c(w_3) = 0 \end{cases}$$

With $C[D](x)$ as the number of times x occurs in the training corpus D . Unknown n-grams are less of a problem in class-based models as in traditional n-gram models, as the number of possible n-grams is reduced drastically. Knowing that we apply an MLE approach, which assigns zero probability to unknown events. The defined morphology model will only outperform the Kneser-Ney model when the sequences $w_1 \dots w_n$ and $w_1 \dots w_{n-1}$ do not occur in the training corpus or occur rarely. It is important to point out that even if $w_1 \dots w_n$ is unknown, $c_1 \dots c_n$ still may be known. We interpolate between the two models to combine the high precision of the Kneser-Ney model with the good generalization capabilities of the morphological model. As we have seen in section 4.1.4, the interpolation parameters λ depend on the entire history of the n-gram at hand:

$$P(w_i|w_{i-N+1}^{i-1}) = (1 - \lambda(w_{i-N+1}^{i-1})) \cdot P_{KN}(w_i|w_{i-N+1}^{i-1}) + \lambda(w_{i-N+1}^{i-1}) \cdot P_M(w_i|w_{i-N+1}^{i-1}) \quad (4.15)$$

However, to reduce complexity and to prevent over-fitting we make the interpolation parameters depend on the cluster index of the last word in the history.

$$\lambda(w_{i-N+1}^{i-1}) := \lambda(c(w_{i-1}))$$

This simplification allows us to compare the performance of a cluster in the language model with the performance on the clustering criteria of section 3.4, as a high parameter $\lambda(c)$ implicates a good performance of the cluster c . However, small clusters with a low token frequency tend to get over- or underrated as they are susceptible to over-fitting.

4.2.2 Evaluation of the Preliminary Model

set	tokens	types	feature vectors
trn	39,027,486	256,872	632
val	4,897,834	100,798	512
valdev	2,449,705	72,564	466
valtst	2,448,129	72,033	472
tst	4,888,973	100,384	512

Table 4.1: Segmentation of the WSJ corpus

We compare the performance of the described model with a Kneser-Ney model and an interpolated model based on part-of-speech (POS) tags. Actually, the relation between words and POS tags is many-to-many, but we transform it to a many-to-one relation by labeling every word - independent of its context - with its most frequent tag. OOV words are treated equally, even if their word classes would not be known in a real application. The experiments are carried out on a Wall Street Journal (WSJ) corpus which is divided into a training, validation and test set as shown in table 4.1 and the validation set itself is further divided into a development and test part. As mentioned above, the training set is used to learn suffixes, word segmentation and the maximum likelihood n-gram estimates. The validation set is used to estimate the unknown word rate ϵ . The unknown word rate of the validation set, knowing the training set is approximately 0.028. Treetagger [24] was used to tag the entire corpus. In the following we use *valdev* for interpolation parameter estimation and *valtst* for testing. The cluster count ∞ denotes a clustering, where two words are in the same cluster, if and only if their features are identical. That is the finest possible clustering of the feature vectors. The results of the experiments can be found in table 4.2.

KN	c_{POS}	c_{50}	c_{100}	c_{∞}
<i>88.06</i>	87.31	87.96	87.92	87.62

Table 4.2: Perplexity results for the preliminary model.

The first observation is that the unclustered feature vectors (c_∞) are better than any of the morphological clusterings. The best improvement to the Kneser-Ney model is 1%. Surprisingly, the POS model, even though it uses oracle information, is only slightly better than our model. However, the improvement is too low to speak of a significantly better performance. In the following section we perform a modification to our model that yields a more noticeable benefit.

4.2.3 The Final Model

In the final model we only cluster words into morphological classes with a token frequency lesser than or equal to a threshold θ . The intuition behind this approach is twofold. As discussed before, a morphological clustering is not able to cover all the irregularities of a language. E.g. it is hard to learn irregular plural forms (e.g. “children” and “geese”) or verb forms (e.g. “struck”, “done”). However, low frequent words tend to regular forms. That is, by introducing the frequency threshold we reduce the difficulty of the clustering problem. Furthermore, we reduce the complexity of the POS distribution. In table 4.3, we compare the tag distributions of words of a frequency of one and zero (out-of-vocabularies) with the token distribution of an entire English corpus.

tag	types		tokens
	$f = 1$	$f = 0$ (OOV)	
Cardinal numbers	0.39	0.38	0.05
Proper nouns	0.35	0.35	0.14
Nouns	0.10	0.10	0.17
Plural nouns	0.05	0.06	0.07
Adjectives	0.05	0.06	0.07
Verbs	0.04	0.05	0.15
Σ	0.98	0.99	0.66

Table 4.3: Proportion of dominant POS for types with training set frequencies $f \in \{0, 1\}$ and for tokens. V^* consists of all verb POS tags.

For low frequencies, cardinal numbers and proper nouns make up more than two thirds of the entire distribution. That is fortunate, as our features are very precise at identifying those two word classes. Additionally the total number of relevant word classes is low. We can conclude that by reducing the frequencies taken into account, we simplify the clustering problem and make our morphological classes more representative to OOVs.

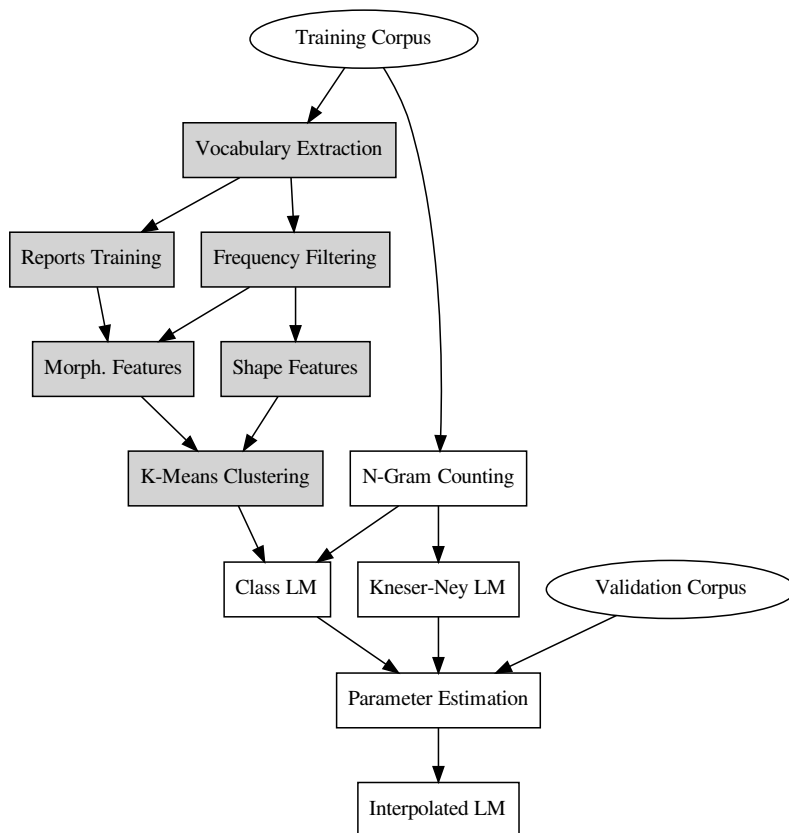


Figure 4.2: A block diagram of the training procedure of the final interpolated model. Gray nodes are part of the cluster process.

We build the new morphological model as follows. Given a training corpus, all words with a frequency below or equal to θ are partitioned into k clusters. The formerly removed high-frequency words are added as singletons to produce a complete clustering. However, OOV words can only be assigned to the original k-means clusters. On account of this, we have to extend the word emission probability to distinguish between singletons and morphological clusters. The probability $P(w_3|c_3)$ is 1 for singleton clusters, is the unknown word rate ϵ if w_3 is unknown and $(1 - \epsilon)$ times the maximum likelihood estimate of w_3 given c_3 else. A diagram of the complete training procedure can be seen in 4.2.

$$p(w_3|c_3) = \begin{cases} 1 & \text{if } c(w_3) > \theta \\ (1 - \epsilon) \frac{c(w_3)}{\sum_{w \in c_3} c(w)} & \text{if } \theta \geq c(w_3) > 0 \\ \epsilon & \text{if } c(w_3) = 0 \end{cases} \quad (4.16)$$

For the interpolation, we use the parametrization of the anterior section, i.e. we introduce one parameter for every cluster. This, however, has one drawback: the number of clusters is now given by the sum of morphological clusters, which is bounded by the number of unique feature vectors (615) and singletons, which is only bounded by the number of word types (256,872). Accordingly, especially for small values of θ , we get a high number of parameters and the model tends to overfit.

4.2.4 Evaluation of the Final Model

We repeat the experiments of the prior section for different values of the threshold parameter θ . A θ -value of ∞ denotes a model, where no words are excluded from the morphological clustering, i.e. the preliminary model. On the other hand, $\theta = 0$ corresponds to the uninterpolated Kneser-Ney model. Again we use valdev to estimate optimal interpolation parameters and calculate the perplexities on *valtst*.

θ	c_{POS}	c_1	c_{50}	c_{100}	c_∞
0	88.06	88.06	88.06	88.06	88.06
1	89.74	89.84	89.73	89.74	89.74
5	89.07	89.36	89.07	89.06	89.07
10	88.59	89.01	88.58	88.57	88.58
50	86.72	87.58	86.69	86.68	86.68
10^2	85.92	87.06	85.92	85.91	85.89
10^3	84.43	86.88	84.83	84.77	84.56
10^4	85.22	87.59	85.89	85.73	85.26
10^5	86.82	87.99	87.44	87.32	86.79
∞	87.31	88.06	87.96	87.92	87.62

Table 4.4: Perplexities over all events of *valtst*.

Table 4.4 shows the results of our experiments. The KN model yields a perplexity of 88.06 on *valtst* (top row). For small frequency thresholds overfitting effects cause that the interpolated models are worse than the KN model. We can see that a clustering of the feature vectors is not necessary, as the differences between all cluster models are small and c_∞ is the overall best model. We can confirm again that morphological clustering and POS classes are close even though the POS class model uses oracle information to assign the right POS to an unknown word. The optimal threshold is $\theta = 10^3$ – the bolded perplexity values 84.43 and 84.56; that means that only 1.35% of the word types were excluded from the morphological clustering

(86% of the tokens). The improvement over the KN model is 4%. As a final experiment, we evaluated our method on the test set. In this case, we used the entire validation set for parameter tuning (i.e., valdev and valtst). The overall perplexity of the KN model is 88.28, the perplexities for the best POS and c_∞ cluster model for $\theta = 1000$ are 84.59 and 84.71 respectively, which corresponds again to an improvement of 4%. The complete results are listed in table 4.5.

θ	c_{POS}	c_1	c_{50}	c_{100}	c_∞
0	88.28	88.28	88.28	88.28	88.28
1	89.03	89.14	89.03	89.03	89.03
5	88.35	88.63	88.34	88.33	88.33
10	87.86	88.26	87.84	87.83	87.82
50	86.39	87.24	86.35	86.34	86.33
10^2	85.90	87.04	85.89	85.88	85.85
10^3	84.59	87.07	84.99	84.92	84.71
10^4	85.42	87.81	86.08	85.92	85.43
10^5	87.03	88.21	87.65	87.52	86.98
∞	87.53	88.28	88.18	88.15	87.84

Table 4.5: Perplexities over all events of the test set.

Performance on Out-Of-Vocabulary Words

As we pointed out in the introduction, the practical problem we try to solve are OOVs. OOVs cause two kinds of problems. Firstly, it is harder to predict words after histories that contain OOVs. In such a case, a back-off model has to reduce the applied history length. In the worst case only the unigram estimate can be used.

$$P(w_3|w_1w_{OOV}) \rightarrow P(w_3)$$

$$P(w_3|w_{OOV}w_2) \rightarrow P(w_3|w_2)$$

We therefore do a second evaluation of our experiments. We reduce the perplexity calculations to events, where the history contains unknown words. This is still a fair evaluation as we always select all the continuations of a selected history.

θ	c_{POS}	c_1	c_{50}	c_{100}	c_∞
0	813.50	813.50	813.50	813.50	813.50
1	181.25	206.17	182.78	183.62	184.43
5	152.51	185.54	154.52	152.98	153.83
10	147.48	186.12	149.34	147.98	147.48
50	146.21	203.10	142.21	140.67	140.46
10^2	149.06	215.54	143.95	142.48	141.67
10^3	173.91	279.02	164.22	159.04	150.13
10^4	239.72	349.54	221.42	208.85	180.57
10^5	317.13	373.98	318.04	297.18	236.90
∞	348.76	378.38	366.92	357.80	292.34

Table 4.6: Perplexities over all events of valstst, where w_1 or w_2 are unknown.

The results can be found in table 4.6. The perplexity for the KN model is 813.50 (top row). A first observation is that the perplexity of model c_1 starts at a good value, but worsens with rising values for $\theta \geq 10$. The reason is the dominance of proper nouns and cardinal numbers at a frequency threshold of one and in the distribution of OOV words (cf. Table 4.3). The c_1 model with $\theta = 1$ is specialized for predicting words after unknown nouns and cardinal numbers and two thirds of the unknown words are of exactly that type. However, with rising θ , other word classes get a higher influence and different probability distributions are superimposed. The best morphological model c_∞ reduces the KN perplexity of 813.50 to 140.46 (bolded), an improvement of 83%. If we repeat the experiment on the test set we get the results in 4.7.

θ	c_{POS}	c_1	c_{50}	c_{100}	c_∞
0	767.25	767.25	767.25	767.25	767.25
1	181.94	207.12	182.52	182.91	181.83
5	156.42	189.47	157.47	155.24	155.72
10	152.67	190.33	152.98	151.90	150.71
50	150.90	207.64	146.78	145.00	144.77
10^2	154.01	221.07	148.28	146.91	145.51
10^3	179.71	286.88	169.79	164.69	156.14
10^4	247.51	359.89	230.65	217.41	187.60
10^5	325.76	384.86	327.67	306.52	244.80
∞	357.96	389.44	377.93	368.57	301.56

Table 4.7: Perplexities over all events of the test set, where w_1 or w_2 are unknown.

The KN model perplexity is 767.25 and the POS and c_∞ cluster model

perplexities at $\theta = 50$ are 150.90 and 144.77. Thus, the morphological model reduces perplexity by 81% compared to the KN model.

The second problem is related to the prediction of OOV words. The standard approach to turn a closed vocabulary language model into an open vocabulary is to treat all unknown words as occurrences of the same unknown word token. As in the word emission probability of our class-based model the unknown word rate can then be discounted from seen events and can be redistributed to unknown words.

$$P_{open}(w_3|w_1w_2) = \begin{cases} (1 - \epsilon) \cdot P_{closed}(w_3|w_1w_2) & \text{if } c(w_3) > 0 \\ \epsilon & \text{if } c(w_3) = 0 \end{cases} \quad (4.17)$$

Unfortunately, it is much harder to evaluate the improvement in the unknown word prediction, because the events where w_3 is unknown cannot be isolated, as any calculation that does not select all continuations of one history would be arbitrary. The resulting perplexity could be easily minimized by a model that assigned 1 to unknown continuations and 0 to known continuations. That, however, would be a terrible model. So we propose a different experiment. We select all events where w_3 is an unknown word but for every selected trigram we also select all trigrams that start with the same history. Following this approach we select 472,746 of the 2,550,627 trigrams of the test set. Table 4.8 shows the results of this experiment for the best performing morphological model c_∞ and the POS model.

	0	1	10	10 ²	10 ³	10 ⁴	10 ⁵	∞
c_{POS}	<i>325.00</i>	323.69	319.77	316.02	314.10	318.19	321.75	323.20
c_∞	<i>325.00</i>	323.56	319.33	315.21	313.47	317.16	320.61	324.05

Table 4.8: Perplexities over histories of valtst that are continued by OOV words for different thresholds.

	0	1	10	10 ²	10 ³	10 ⁴	10 ⁵	∞
c_{POS}	<i>330.91</i>	329.58	325.78	321.53	318.95	323.04	327.22	328.73
c_∞	<i>330.91</i>	330.38	325.33	320.82	318.65	322.34	326.47	330.02

Table 4.9: Perplexities over histories of tst that are continued by OOV words for different thresholds.

The results of this experiments show the same behavior as the overall perplexity results, i.e. the best model c_∞ achieves at a θ of 1000 an improvement of approximately 3.5%. The perplexities of the uninterpolated

Kneser-Ney model can be found again at a θ of 0. Again table 4.9 shows the results for the same experiment on the *tst* corpus. The improvement achieved by the morphological model is approximately 3.7%.

Evaluating the Morphological Clusters

As said in section 4.2.1, we can use the estimated interpolation parameters $\lambda(c(w_2))$ to evaluate the performance of a specific cluster. It is important to point out though that high parameter values only indicate good clusters. Especially small clusters suffer from overfitting and statistical errors. Nevertheless, we will use this evaluation method to shed some light on the relation between POS tags, suffixes and language model performance. For a parameter value of 0, the interpolated model performs exactly as the Kneser-Ney model² a value of 1 corresponds equally to the morphological model. We already know that the high precision of the Kneser-Ney model is hard to beat, so we expect the values of λ to lie below 0.5 for most clusters (or even close to 0). In the following, we are going to compare the best clusters of the POS model and the morphological models at $\theta = 1000$. We start with the POS model, whose 20 best performing clusters are listed in table 4.10.

The maximal value of λ in this list is 0.49 that means even the best cluster is slightly worse than the Kneser-Ney model. The reason is that in any cluster we find subsets that vary in their usage. An example in the cardinal number cluster are small amounts of money or year numbers. This over-generalization is also the reason, why we can achieve better clusters using the morphological approach. In the next tables, we find clusters of a model that is only based on the suffix feature (S_1). Cf. table 4.11 for the 20 best clusters.

The best of these suffixes like e.g. “-ville” and “-tec” share the property that they restrict the words to a common word class (here proper nouns), but also to a common sense. The suffix “-ville” for example is French and means city and for historical reasons a lot of Canadian and United States cities have a name that ends with this suffix. The suffixes “-tec” and “-tek” are modern suffixes that can be found in many technology related company names. Other suffixes in the list like e.g. “-ers” are actually two concatenated suffixes that the report algorithm happens to learn as one suffix. In this case that is very fortunate as “-s” would be a very general suffix that is even frequent in two word classes. However, the “-ers” cluster in our model groups mostly persons together. Table 4.12 lists ten more common English suffixes, to demonstrate that they are not the most valuable suffixes for a morphological language

²One might say the interpolated model is the Kneser-Ney model

POS	λ	Word Types	Examples
CD	0.49	71042	0.5 200,000 82 125 66 4.2 350 89 1.25 0.
NNS	0.36	17927	fields classes ships 1960s posts passeng
RB	0.26	3274	temporarily repeatedly maybe therefore p
JJ	0.24	15053	numerous overnight tiny religious precio
NP	0.22	96154	Northrop III Midwest Southwest Gordon Ke
VVN	0.22	4707	convicted drawn registered founded deliv
NN	0.21	29600	consideration guy leasing theater import
PP	0.21	16	ourselves myself yourself herself ours t
JJS	0.20	354	greatest Best strongest fastest nearest
VV	0.19	2871	oppose sit speak recover occur enable de
JJR	0.19	438	harder younger cheaper slower tougher wi
VVG	0.18	6410	hoping causing watching serving underlyi
MD	0.15	11	ought Can Will Should shall Would Could
SYM	0.14	11	z x e d] r f = _ \
VVZ	0.14	2695	promises plays expires pays helps keeps
NPS	0.13	461	Airways Brands Philippines Electronics N
VVD	0.12	1716	insisted asserted marked stated narrowed
DT	0.12	11	Any Every Half Either AN NO ALL THIS SOM
UH	0.12	21	Yes yes Oh Please alas Alas Welcome Yeah
WRB	0.07	10	Where whenever wherever Whenever whereby

Table 4.10: List of the 20 best POS-classes for the c_{POS} model at $\theta = 1000$.

λ	Word Types	Examples
0.40	247	Nashville Louisville Jacksonville Greenville
0.37	77	Vortec Memotec Portec Vasotec Equitec
0.36	61	Nortek Armtek Viratek Ametek Wavetek
0.29	1212	businessman Rudman Freeman Norman Human
0.28	462	optimism skepticism terrorism mechanism
0.28	70	Becor Encor Centocor Balcor Fidelcor
0.27	1577	foreigners advertisers barriers bidders
0.26	453	Hawley Staley Bradley Berkeley Bailey
0.25	72	accountability respectability marketability
0.25	688	Thompson Robertson Anderson Donaldson
0.24	69	Textron Vestron Quotron Electron Vernitron
0.23	94	GenCorp Unicorp Equiticorp PacifiCorp SCEcorp
0.23	755	willingness competitiveness illness witness
0.23	23555	fields promises classes Miss ships plays Airways
0.22	185	readiness happiness uneasiness unhappiness
0.22	2735	repeatedly politically suddenly completely
0.20	63	bureau plateau Trudeau Chateau Comeau Rousseau
0.19	126	Richfield Fairfield Springfield battlefield
0.19	286	WHO'S CORP.'S IT'S PLC'S CO.'S REAGAN'S INC.'S
0.18	105	Genentech Wedtech KaiserTech Ameritech Sematech

Table 4.11: The 20 best clusters of the c_∞ model at $\theta = 1000$, using only the suffix features.

λ	Word Types	Examples
0.23	755	willingness competitiveness illness witness
0.23	23555	fields promises classes Miss ships plays Airways
0.22	2735	repeatedly politically suddenly completely
0.17	7100	insisted convicted registered founded asserted
0.15	8032	leasing hoping causing watching serving controlling
0.11	330	regardless jobless homeless Wireless endless
0.08	429	initiative extensive defensive survive excessive
0.07	1416	Occidental withdrawal Environmental rental
0.05	214	importance compliance alliance appearance
0.02	189	dislike businesslike childlike lifelike lookalike

Table 4.12: List of more common suffixes for the same model as in 4.11.

model.

At first it is a bit irritating that the λ of the “-s” cluster is so high. However, an analysis shows that it is dominated by a plural proper noun cluster that as just seen in table 4.10 does perform well. The other clusters, even though related to a specific word class, only produce moderate results. In the last table 4.13, the 20 best clusters of the c_∞ model can be found.

The table points out how the additional shape-like features improve the clustering generated by the suffixes. A very good example is the cluster of Canadian and Australian dollar amounts.

λ	Word Types	Examples
0.86	1831	C\$100 C\$75 C\$50 C\$200 C\$10 C\$30 C\$53 A\$100
0.73	712	12:01 9:30 10:30 8:30 7:30 3:30 11:30 2:30
0.70	1226	401(k 2036(c 1,000th a.357 a.38 b9.76 10(b
0.70	85	0.5 4.2 0.2 3.8 2.9 3.3 6.5 3.6 0.4 3.2 0.3
0.64	351	No.1 Feb.245 Sept.30 No.2 Jan.1 Dec.31 Apr.'87
0.60	69201	200,000 1.25 20,000 3,000 300,000 30,000 250,000
0.58	901	125 350 110 180 140 130 450 750 900 160 225
0.46	1438	1969 1977 1976 1997 2000 1973 1975 1998 1994
0.43	240	Nashville Louisville Jacksonville Greenville
0.39	435	TRADING ADVERTISING LOSING WINNING PUBLISHING
0.39	86	Georgetown Morristown Tarrytown Youngstown
0.37	53	Successful Beautiful Fearful Wonderful Useful
0.36	57	Nortek Armtek Viratek Ametek Wavetek Howtek
0.35	72	Vortec Memotec Portec Vasotec Equitec Thortec
0.35	923	Rudman Norman Kaufman Grumman Friedman Eastman
0.35	53	Trudeau Comeau Rousseau Loiseau Martineau
0.33	286	WHO'S CORP.'S IT'S PLC'S CO.'S REAGAN'S INC.'S
0.32	52	Southmark Denmark Equimark Raymark Intermark
0.31	57120	Northrop Gordon Kenneth Pillsbury Newmont
0.31	89	Tourism Journalism Prism Criticism Semitism
0.30	3884	NATO NASA NASD AMEX CFTC NCNB OSHA RICO LONDON

Table 4.13: The 20 best clusters of the c_∞ model at $\theta = 1000$, using the entire feature vector.

Chapter 5

Conclusion

We presented a new class-based morphological language model. The model uses morphological and shape-like features, which can be trained on complete unannotated corpora. We evaluated these features for their ability to generate word clusters similar to part-of-speech (POS) classes. However, we demonstrated that they also lead to a number of precise semantic clusters, such as city names, company names and money amounts. In a number of experiments the model outperformed a state-of-the-art modified Kneser-Ney model in general by 4% and even by 81% in the prediction of the continuations of histories containing OOV words. As said, the model is entirely unsupervised, but works as well as a model using part-of-speech information and oracle information to assign POS tags to out-of-vocabulary words. This demonstrates that even for English, which usually has a low unknown word rate and a simple morphology, morphological word modeling is worthwhile and should be further investigated in the future.

Chapter 6

Future Work

We plan to test the domain adaptation properties of our model and how it performs in NLP applications such as machine translation or speech recognition. Furthermore, we want to extend our model to other languages such as German and Spanish. This could be more challenging, as certain languages have a more complex morphology – e.g. Spanish suffixes are more ambiguous than English suffixes – but also worthwhile, as the unknown word rate in these languages is higher. A last interesting question is, how the language model performance scales with the corpus size. Even with the availability of huge English corpora nowadays, this still remains important as many NLP tasks require specialized data sets. An example are oral corpora in Speech Recognition.

The model could be further improved by using contextual information for the word clustering and training a classifier based on morphological features to assign OOV words to these clusters. Another enhancement could be achieved by using the `suffix_stem` feature to split big clusters, such as “-s”, “-ed” or “-ing”.

Appendix A – Proof of the Equality of Normalized Mutual Information and V_1 -Measure

The original proof and the basic idea of checking for equality are from Hamidreza Kobdani.

For $H(X, Y) = 0$ the mutual information $I(X, Y)$ equals $H(X) + H(Y)$ and thus the normalized mutual information equals 1, so does the V-Measure by definition. For the case $H(X, Y) \neq 0$, we show the equality by expressing homogeneity and completeness in terms of mutual information:

$$I(X, Y) = H(X, Y) - H(X|Y) - H(Y|X)$$

$$h = 1 - \frac{H(X|Y)}{H(X)} = \frac{I(X, Y)}{H(X)}$$

$$c = 1 - \frac{H(Y|X)}{H(Y)} = \frac{I(X, Y)}{H(Y)}$$

$$\begin{aligned} V_1 &= \frac{2 \cdot h \cdot c}{h + c} \\ &= \frac{2 \cdot \frac{I(X, Y)^2}{H(X)H(Y)}}{\frac{I(X, Y)}{H(X)} + \frac{I(X, Y)}{H(Y)}} \\ &= \frac{2 \cdot I^2(X, Y)}{H(Y) \cdot I(X, Y) + H(X) \cdot I(X, Y)} \\ &= \frac{2 \cdot I(X, Y)}{H(Y) + H(X)} \\ &= NI \end{aligned}$$

Appendix B – The Penn Treebank Tagset

The following table contains the tags used in this article. The Penn tagset used by Treetagger is actually a slight modification of the original Penn treebank tagset. Treetagger distinguishes between the verb forms VH (form of to have), VB (form of to be) and VV (form of another verb). A complete description can be found in [16].

Tag	Description
CD	Cardinal number
DT	Determiner
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
NN	Noun, singular or mass
NNS	Noun, plural
NP	Proper noun, singular
NPS	Proper noun, plural
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
VV	Verb, base form
VVD	Verb, past tense
VVG	Verb, gerund or present participle
VVN	Verb, past participle
VVP	Verb, non-3rd person singular present
VVZ	Verb, 3rd person singular present

Bibliography

- [1] Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, Robert L. Mercer, and David Nahamoo. A fast algorithm for deleted interpolation. In *2nd European Conference on Speech Communication and Technology*, pages 1209–1212, 1991.
- [2] Andre Berton, Pablo Fetter, and Peter Regel-Brietzmann. Compound words in large-vocabulary German speech recognition systems. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 2, pages 1165 –1168 vol.2, October 1996.
- [3] Jeff A. Bilmes and Katrin Kirchhoff. Factored language models and generalized parallel backoff. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers - Volume 2*, NAACL '03, pages 4–6. Association for Computational Linguistics, 2003.
- [4] Peter F. Brown, Peter V. de Souza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18:467–479, December 1992.
- [5] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ACL '96, pages 310–318. Association for Computational Linguistics, 1996.
- [6] Carlos Crespo, Daniel Tapias, Gregorio Escalada, and Jorge Alvarez. Language model adaptation for conversational speech recognition using automatically tagged pseudo-morphological classes. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 2, pages 823 –826 vol.2, April 1997.
- [7] Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar,

- and Andreas Stolcke. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Trans. Speech Lang. Process.*, 5:3:1–3:29, December 2007.
- [8] Vera Demberg. A language-independent unsupervised model for morphological segmentation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 920–927. Association for Computational Linguistics, June 2007.
- [9] Byron E. Dom. An information-theoretic external cluster-validity measure. Technical report, Research Report RJ 10219, IBM, 2001.
- [10] Antoine Ghaoui, François Yvon, Chafik Mokbel, and Gérard Chollet. Morphology-based language modeling for Arabic speech recognition. In *9th European Conference on Speech Communication and Technology*, pages 1281–1284, 2005.
- [11] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.
- [12] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2 edition, 2008.
- [13] Samarth Keshava. A simpler, intuitive approach to morpheme induction. In *PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*, pages 31–35, 2006.
- [14] Stuart P. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129 – 137, March 1982.
- [15] Giulio Maltese and Federico Mancini. An automatic technique to include grammatical and morphological information in a trigram-based statistical language model. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 157 –160 vol.1, March 1992.
- [16] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: the penn treebank. *Comput. Linguist.*, 19:313–330, June 1993.
- [17] Marina Meilă. Comparing clusterings—an information based distance. *J. Multivar. Anal.*, 98:873–895, May 2007.

- [18] Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8(1):1–38, 1994.
- [19] Thomas R. Niesler, Edward W.D. Whittaker, and Philip C. Woodland. Comparison of part-of-speech and automatically derived category-based language models for speech recognition. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 1, pages 177–180 vol.1, May 1998.
- [20] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):pp. 846–850, 1971.
- [21] Roi Reichart and Ari Rappoport. The nvi clustering evaluation measure. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning, CoNLL '09*, pages 165–173. Association for Computational Linguistics, 2009.
- [22] Andrew Roseberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure, 2007.
- [23] Roni Rosenfeld. A Maximum Entropy Approach to Adaptive Statistical Language Modeling, 1996.
- [24] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, 1994.
- [25] Hinrich Schuetze and Michael Walsh. Half-context language modeling. Unpublished manuscript, 2010.
- [26] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.
- [27] Ulla Uebler and Heinrich Niemann. Morphological modeling of word classes for language models. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP 1998)*, 1998.
- [28] Dimitra Vergyri, Katrin Kirchhoff, Kevin Duh, and Andreas Stolcke. Morphology-based language modeling for Arabic speech recognition. In *8th International Conference on Spoken Language Processing*, pages 2245–2248, 2004.

- [29] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2. edition, 2005.
- [30] Deniz Yuret and Ergun Biçici. Modeling morphologically rich languages using split words and unstructured dependencies. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACL-IJCNLP '09, pages 345–348. Association for Computational Linguistics, 2009.