

# NATURAL LANGUAGE PROCESSING OF LARGE PARALLEL CORPORA

**Dániel Varga**

**Ph.D. Dissertation**

**Supervisors:**

András Kornai D.Sc., András Lukács Ph.D.



Eötvös Loránd University  
Faculty of Informatics  
Department of Information Systems

Ph.D. School of Computer Science  
András Benczúr D.Sc.

Foundations and Methods of Informatics Ph.D Program  
János Demetrovics D.Sc. MHAS

Budapest, 2012

# Contents

	Page
1 Introduction . . . . .	1
2 Mathematical preliminaries . . . . .	6
2.1 Supervised learning . . . . .	7
2.2 Maximum entropy modeling . . . . .	9
2.3 Sequence labeling . . . . .	15
3 Morphological Analysis . . . . .	22
3.1 The hunmorph annotation formalism . . . . .	22
3.2 Tools - <b>hunlex</b> , <b>hunmorph</b> . . . . .	31
3.3 Resource built - morphdb.hu . . . . .	35
3.4 Applications . . . . .	37
4 Morphological Disambiguation . . . . .	40
4.1 Background . . . . .	41
4.2 Resource used - the Szeged Corpus . . . . .	42
4.3 Related work . . . . .	44
4.4 The hardness of the tagging task . . . . .	45
4.5 Our algorithms . . . . .	49
4.6 Evaluation . . . . .	52
4.7 Later work . . . . .	54
4.8 Resource built - the Szószablya corpus and lemmatized frequency dictionary . . . . .	55
4.9 Applications . . . . .	58

5	Named Entity Recognition . . . . .	67
5.1	Background . . . . .	68
5.2	Resources used . . . . .	69
5.3	Algorithm - <b>hunner</b> . . . . .	72
5.4	Evaluation . . . . .	77
5.5	Metonymy resolution . . . . .	79
5.6	Resources and applications . . . . .	80
6	Noun phrase chunking . . . . .	83
6.1	Background . . . . .	83
6.2	Resource used - the Szeged Treebank . . . . .	85
6.3	Algorithm - <b>hunchunk</b> . . . . .	87
6.4	Evaluation . . . . .	88
6.5	Later work and applications . . . . .	90
7	Sentence Alignment . . . . .	92
7.1	Background . . . . .	92
7.2	Algorithm - <b>hunalign</b> . . . . .	95
7.3	Algorithm - <b>partialAlign</b> . . . . .	107
7.4	Resource built - the Hunglish Corpus . . . . .	108
7.5	Resource built - the JRC-Acquis Corpus . . . . .	115
7.6	Later work . . . . .	117
7.7	Applications . . . . .	118
8	The Hunglish bitext query system . . . . .	120
8.1	The <b>huntools</b> multilingual text processing framework . . . . .	121
8.2	The Hunglish bitext query system . . . . .	126
	Bibliography . . . . .	134

# Chapter 1

## Introduction

Computational Linguistics (CL) is collective work, and the systems that catch the imagination, from Google Translate (machine translation) to IBM’s Watson (question answering) are the work of teams with hundreds of people. The reason for this is that human language is tremendously complex, and even the major subsystems are too complex for the individual researcher to grasp.

The reader expecting to see here an in-depth study of some carefully selected phenomenon will be disappointed. Our primary goal is breadth (known in CL as *coverage* or *recall*), rather than depth. We see our work as *enabler* of in-depth research of linguistic phenomena. *First* we need to create corpora, and the automatic tools to analyze them, and *only then* can we look at such phenomena in depth. This thesis describes work devoted to these and similar *low-level* tasks. We firmly believe that modern linguistics can benefit tremendously from the kind of infrastructure we set out to create.

That the low-level work has significant payoff in higher-level research will be seen from our discussion of several studies (Szeredi 2009), (Prekopcsák 2008), (Miháltz & Pohl 2006), some coauthored by the author (Pléh et al. 2011) where the key low-level computational and data resources are true enablers, without which the study could not have gotten off the ground. Further, the experience of the past two decades has shown that in-depth studies where the data was collected manually, primarily by relying on the intuitions of the linguist, are surprisingly *fragile*, in that new data that escaped the attention of the original authors will almost always break the framework constructed on limited data. A typical example would be pronunciation dictionaries such as (Kenyon & Knott 1953), where in actuality such

a small fraction of the pronunciation alternatives heard in connected speech are listed that a speech recognition system strictly relying on this data would fail to recognize two thirds of the words.

To avoid such pitfalls, CL adopted a highly statistical methodology. Various algorithms attempting to solve the same problem are measured against each other based on their performance on random *test data*, carefully controlling for factors such as access to known solutions (known as *training data*). As we shall discuss in Chapter 2, statistical methods have penetrated the field far more deeply than statistics-based evaluation. In fact, most CL algorithms used today are either purely statistical *machine learners*, or hybrids combining rule-based and statistical components. In later chapters we will see examples of both, but we note here in advance that ours is not a thesis in Machine Learning (ML) proper, in that the emphasis is on actually working, fast, and scalable CL systems, rather than on their ML core. A key consideration, very much necessitated by the collective nature of the work, was to create Free/Libre Open Source Software that can be reused by other researchers without license, patent, or copyright worries, as indeed we have relied extensively on free software and data created by other teams.

Chapter 3, dealing with morphology, draws the line halfway across morphological analysis, treating the inflectional and derivational parts of the problem separately. The reason is that for most applications, ranging from information retrieval to machine translation, it is only the inflectional affixes (which are productive, category-preserving, and which leave the meaning of the stem intact) that need to be found, and derivational affixes (which are unsystematic, and often change both category and meaning) are irrelevant.

Chapter 4 deals with the problem of morphological disambiguation. An analysis system is by definition behaving correctly if it returns two analyses for *fent* corresponding to the meanings ‘sharpened’ and ‘up’, but in a given context such as *ollót fent* ‘he sharpened scissors’ and *fent nincs olló* ‘there are no scissors up there’ one of the two must be selected, and the systems we present in this chapter deal with this issue.

Chapter 5 addresses the identification of proper noun phrases or *named entities*. Since much of our knowledge about the world is anchored to places, people, or organizations, finding and correctly identifying parts of text that refer to named entities is key to understanding natural language input. This task is called Named Entity Recognition (NER). NER has its own disambiguation problems, like detecting and classifying *metonymic* readings such as place-for-event ‘*Vietnam was a national trauma.*’. Section 5.5 discusses this subproblem briefly, but the emphasis of the chapter is on our **hunner** named entity recognition system for Hungarian.

Chapter 6 deals with the task of *noun phrase chunking*. High accuracy analysis of full syntactic structure, assigning a parse tree or other structural description to a sentence, is beyond the current state of the art. Fortunately, for many applications it is often sufficient to detect NPs (noun phrases) in a sentence. NP chunking of the kind performed by the **hunchunk** system is also a stepping stone toward full syntactic analysis, in part because analysis below the NP level can be performed by standard tools such as Context Free Grammars (Abney 1991) and in part because by bracketing the NPs higher level analysis is greatly facilitated.

Chapter 7 deals with *sentence alignment*, a low-level task that is key not just to the modern methods of machine translation but also to automated dictionary and thesaurus building. Our **hunalign** system performing this task is distinguished from the alternatives mainly by performing, in the critical range of interest, an order of magnitude faster, without sacrificing accuracy. We present a long list of Hungarian and international corpus creation efforts and other applications of our tools, detailing those where the author was a participant.

Finally, Chapter 8 deals with the **hunglish** bitext query system, which brings the ‘crowd-sourcing’ paradigm to the problem of creating and validating CL data. Neither Chapter 2, which presents the theoretical background in machine learning, nor this last chapter fits entirely the presentation structure we use throughout the body of the thesis:

1. Definition of a natural language processing task.
2. Review of earlier work on this task.

hunmorph	<a href="http://mokk.bme.hu/resources/hunmorph">http://mokk.bme.hu/resources/hunmorph</a>
morphdb.hu	<a href="http://mokk.bme.hu/resources/morphdb-hu">http://mokk.bme.hu/resources/morphdb-hu</a>
huntoken	<a href="http://mokk.bme.hu/resources/huntoken">http://mokk.bme.hu/resources/huntoken</a>
hunpos	<a href="http://code.google.com/p/hunpos/">http://code.google.com/p/hunpos/</a>
webcorpus pipeline	<a href="https://github.com/zseder/webcorpus">https://github.com/zseder/webcorpus</a>
hunner, hunchunk	<a href="https://github.com/recski/HunTag.git">https://github.com/recski/HunTag.git</a>
hunalign	<a href="http://mokk.bme.hu/resources/hunalign">http://mokk.bme.hu/resources/hunalign</a>
Hunglish bitext query	<a href="http://code.google.com/p/hunglish-webapp">http://code.google.com/p/hunglish-webapp</a>
Szószablya Webcorpus	<a href="http://mokk.bme.hu/resources/webcorpus">http://mokk.bme.hu/resources/webcorpus</a>
Frequency Dictionaries	<a href="http://hlt.sztaki.hu/resources/webcorpora.html">http://hlt.sztaki.hu/resources/webcorpora.html</a>
Hunglish Corpus	<a href="http://mokk.bme.hu/resources/hunglishcorpus">http://mokk.bme.hu/resources/hunglishcorpus</a>
JRC-Acquis Corpus	<a href="http://langtech.jrc.it/JRC-Acquis.html">http://langtech.jrc.it/JRC-Acquis.html</a>
Szószablya dictionary service	<a href="http://szotar.mokk.bme.hu/szoszablya">http://szotar.mokk.bme.hu/szoszablya</a>
Hunglish bitext query service	<a href="http://hunglish.hu">http://hunglish.hu</a>

Table 1.1: Our tools and resources.

3. Description of the source and/or construction of our training data where our solution is based on supervised learning.
4. Presentation of an algorithm for the task.
5. Evaluation of the algorithm on a test corpus, comparison of the results to the state-of-the-art when possible.
6. Description of an automatically constructed corpus that we have built using the tool.
7. Presentation of our own applications based on this corpus.
8. Review of further literature on existing applications of our tool and corpus.

On Table 1.1 we list the most important software tools, language resources and services created with the author's participation. (The author had minor to no role in the creation of the first four items on the list, we provide them for the sake of completeness, as they are important components of our language processing toolchain.)

If there is a central theme binding together this thesis it is the collaborative nature of the CL undertaking, manifested not just in the free exchange of ideas, software, and data, but also in the incremental development of methods.

The work was performed nearly always in teams, of which we single out the Budapest Institute of Technology Media Research Centre (BME MOKK), which provided an ideal research environment. My thanks go first to my two advisors. I am grateful to advisor András Lukács for all his encouragement. Our discussions on machine learning have always been illuminating to me. I would like to express my gratitude towards my advisor András Kornai. If there are any good ideas in this dissertation, chances are they are his. Without him, this thesis simply would not have been completed.

My thanks go to all my co-authors. I am especially grateful to co-author Gábor Recski, who provided invaluable help and insight. A large number of people helped me with proof-reading. The usual disclaimer stating “all remaining errors are mine” is even more important than usual, for two reasons: first, many of the listed have only seen early drafts or partial versions. Second, I had to disregard many excellent suggestions and criticisms due to the nearing deadline. The list includes Eszter Simon, Viktor Trón, Attila Zséder, Dávid Nemeskey, Márton Makrai, Judit Ács, Katalin Pajkossy, Csaba Holman, Viktor Nagy, Péter Torma, Daniel Burfoot, Jen Runds. I thank Heni, my wife, for all of her support and patience.



## Chapter 2

### Mathematical preliminaries

In this chapter we introduce the machine learning background and terminology necessary for the rest of the thesis. What the chapter covers is standard material, so obviously there are much more systematic and insightful introductions to the concepts discussed below (Bishop 2007), (Jurafsky & Martin 2008), (MacKay 2002), (Rabiner 1989), (Sutton & McCallum 2011). The first aim of the chapter is to introduce machine learning terminology used in the thesis, in plain English, without much formalization that would obscure the main points. Our second goal is to hint at the motivations behind these methods, more with the student of (computational) linguistics than the student of machine learning in mind. Machine learning experts will find the material of the chapter familiar, but we note that natural language processing terminology differs slightly from that of other areas of machine learning.

The first half of the chapter introduces commonly used concepts in supervised learning such as feature vectors, linear classifiers, loss functions, the maximum likelihood method, and regularization. The text introduces these concepts ‘on the fly’, building up the preliminaries gradually to introduce the modeling approach used throughout the thesis: regularized maximum entropy learning.

The second half of the chapter deals with the task of sequence labeling. We introduce Hidden Markov Models, Maximum Entropy Markov Models, and we briefly mention Conditional Random Fields. Many natural language processing tasks can be phrased as an instance of sequence labeling, and we will stay within this paradigm in the thesis when we give algorithms for morphological disambiguation, named entity recognition, and noun phrase chunking.

## 2.1 Supervised learning

In machine learning terminology, supervised learning means learning some function based on labeled data points (the training data or training corpus). The task of a supervised learning method is to infer the function from the training data under some reasonable assumptions about the form of the function, and then successfully predict the value of the function on previously unseen data points. (We will call the second set evaluation data or test data.)

Supervised learning is to be contrasted with unsupervised learning. Here we have no labeled training data, and the task of the unsupervised learning method is to find hidden structure (regularities) within the data.<sup>1</sup> Almost all results of the thesis employ supervised learning techniques, so this is the approach we will discuss in this chapter. The only major exception is our unsupervised **hunalign** sentence alignment algorithm which we will discuss in Chapter 7.

The explicit assumption behind supervised learning is that if the training corpus is a statistical sample of some probability distribution over the input data, then the learner trained on the training corpus can be used to solve some task when given new data that comes from the same probability distribution. The stronger assumption, often left implicit, is that the learner will provide useful output for new input even when the new input comes from a probability distribution that is different, but in some sense close to the distribution of the training data. If our learning method is robust, focusing on general patterns rather than peculiarities of the training data, then we have reason to hope, although no theoretical guarantee, that a learner trained on a corpus of newswire text will degrade gracefully when employed on a transcription of spoken discourse. We talk about overfitting when the learner picks up regularities of the training corpus that are irrelevant or detrimental when it is evaluated on previously unseen data.

---

<sup>1</sup>Often, the two methods are combined, as in semi-supervised learning, where a small amount of (typically expensive to create) labeled training data is augmented with a larger amount of (cheap) unlabeled data. This thesis does not employ semi-supervised techniques.

During supervised learning we allow the training data to be noisy, so we are not surprised if it contains contradictory information about the value of the function on a given input. In natural language processing, the data is often virtually noiseless in a strict technical sense, meaning it is a result of mostly deterministic processes. But the noise tolerance of the learning method helps us mitigate the information loss caused by the limitations of our feature extraction (see below) and modeling.

From here, we will focus on classification, the case when the range of the function to be learned is finite and known in advance. This is a valid assumption for most natural language processing tasks. Depending on context, we will use two synonyms for the values of the function to be learned: we will call them classes or labels.

### 2.1.1 Feature extraction

The training data is a set of points from the input domain, paired with their known labels. Although the input domain can have any sort of exploitable structure in principle, below we will mostly focus on the case of finite-dimensional vector spaces of real numbers. For a specific application, we map the input domain to vectors in such a vector space. We will call this step feature extraction, and we will call the coordinates of this vector space features. Note that we allow the finite dimension of the vector space to be extremely large. For example, in a morphological disambiguation task we might assign a coordinate to each word in the training corpus, setting the value to 1 if the word appears at the token position we consider, and 0 if it does not appear there. This results in very sparse vectors.

In most of our applications the coordinates are 0-1 valued (binary). If this is the case, a sparse representation of a feature vector is the list of 1-valued features. We will identify features with character strings. As a simple example, take some natural language processing application where a feature vector is assigned to each token of the input text. If the `cap` feature is set, this might mean that the given word is capitalized. Sometimes features are

defined in *feature groups*, for example we could use a distinct feature for each possible character trigram of the token. (For example we set the features `tri:the tri:her tri:ere` for the a word (*there*.) We will sometimes abuse terminology slightly, and write feature when in fact we mean a feature group.

### 2.1.2 Linear classifiers

We call a binary (two-class) classification method a linear classifier if its decision is based on the sign of a linear function of its input features. That is, a linear classifier classifies into two classes according to which side of a hyperplane the data point is at. Linear classifiers are a surprisingly strong and versatile class of classifiers. Although they only deal with binary classification, there are several well-known methods to get multi-class classification either by generalizing specific binary classifiers, or by using them as building blocks. For a more detailed treatment see (Bishop 2007).

Let us assume that we are provided with some training corpus of input-output pairs  $(x(i), y(i)), i = 1, \dots, n$ , for the unknown binary (two-valued) function we need to learn, where  $y(i) = \pm 1$ . There is no a priori unique method for us to choose a best linear classifier based on the training data. Indeed, there are several computationally tractable and theoretically well-motivated possibilities to formalize this choice as an optimization problem. One notable choice we will not discuss here is support vector machines (SVMs), for an introduction see (Burges 1998).

## 2.2 Maximum entropy modeling

In this section we will discuss the maximum entropy (maxent) method, the main building block of most of our models throughout the thesis. One of the attractive properties of this method is that there are two, seemingly very different formalizations of the learning task that both lead to this method. The first formalization treats the question as a regression problem, and solves it using the maximum likelihood method. This regression-based approach is called

multinomial logistic regression, and we discuss it in Subsection 2.2.1. The second formalization treats the question as a constrained optimization problem, and relies on the principle of maximum entropy, hence the name. This approach is introduced in Subsection 2.2.2. Our treatment of both approaches is brief. The interested reader should consult e.g. (Pietra et al. 1997).

The maximum entropy method actually solves a problem more general than classification: It finds a probability function  $p(y|x)$  that takes a feature vector  $x$  and gives a probability estimate that  $x$  belongs to class  $y$ . Given such a probability function, we can build a classifier simply by picking the class with the highest probability. In Section 2.3 we will introduce some examples where probability functions – as opposed to pure classifiers – can be used as building blocks of more complex machine learning methods.

Further on,  $x(i)_j$  means the  $j$ th feature of data point  $i$ .

### 2.2.1 Logistic regression

In this subsection we will introduce logistic regression, a method that solves the binary classification task by solving a regression on the probability function. All results in this subsection could easily be generalized from the binary case to the multi-class case. This generalization is called multinomial logistic regression. For the sake of simplicity and clarity we only discuss the binary case below.

The subsection starts by introducing the specific parametric form in which we search for our probability function. We then introduce the maximum likelihood method to turn the search for a well-behaved probability function into an optimization problem. We end the subsection by discussing Maximum A Posteriori estimation, a refinement introduced to prevent overfitting.

Logistic regression belongs to the class of Generalized Linear Models, where we apply some nonlinear transformation to the linear combination of features before fitting it to the

dependent variable. Here, the dependent variable is probability of class membership. Classification by choosing the class with the higher predicted probability leads to a linear classifier.

In the case of logistic regression, we transform the output of some linear function by applying the logistic function  $\frac{1}{1+e^{-t}}$ , and interpret the output as a probability of class membership. Formally, we seek the probability of  $x$  being in class +1 in the form

$$\frac{1}{1 + e^{-\beta x}},$$

where  $\beta$  is a linear function, mapping feature vectors to real numbers. The use of the logistic function seems ad hoc. It is certainly not the only possible choice for this so-called inverse link function. The only immediately obvious requirement for an inverse link function when fitting probabilities is that it should be monotone increasing and converge to 0 and 1 in the infinities. Indeed, several such inverse link functions are in use.<sup>2</sup> Later we will highlight some special properties of the logistic function that make it a natural choice among possible inverse link functions. For now, we note one such property: the logistic function transforms log-odds to probability.<sup>3</sup> Thus  $\beta x$  can be interpreted as the logarithm of the odds of class membership. This means that changing some feature  $x_j$  by a constant results in a constant *multiplicative* change in the predicted odds of class membership.

After defining a parametric form for the probability function, the next step is to specify how to fit the regression to the training data (observations). In the simplest case, we can use the maximum likelihood method. This means finding the model parameters that maximize the probability of the observations, assuming that the observations are independent when conditioned on the model. The joint probability of the conditionally independent training data points takes the form of a product. Maximizing it is equivalent to maximizing the sum of

---

<sup>2</sup>For example, in the case of probit regression, the inverse link function is the cumulative distribution function of the standard normal distribution. This has thinner tails, meaning that extreme outlier data points have bigger influence on model parameters than in the case of logistic regression.

<sup>3</sup>The odds of a probability  $p$  event is  $p/(1-p)$ . The log-odds (also called logit) is simply the logarithm of odds.

conditional log-probabilities of data points. With nothing but simple algebraic manipulations it can be shown that this is equivalent to minimizing the so-called total loss function:

$$\sum_i \kappa(y(i)\beta x(i)),$$

where  $\kappa(t) = \ln(e^{-t} + 1)$  is the logistic loss function. Intuitively,  $\kappa$  is a penalty function, and we are looking for a separating hyperplane  $\beta$  that minimizes the total penalty score over the training data.  $\kappa$  is close to 0 for large positive values, meaning we do not penalize points that fall on the ‘right’ side of the hyperplane.  $\kappa$  is close to the negative identity function for large negative values, so for points on the ‘wrong’ side of the hyperplane we give a penalty score that is approximately equal to the distance from the hyperplane.  $\kappa$  is positive near zero, meaning that the hyperplane is slightly penalized when correctly-classified points are too close to it. Compare this to the so-called hinge loss of Support Vector Machines: in that case the loss function is  $\max(0, 1 - t)$ , a good piece-wise linear approximation of  $\kappa$ . All three above-mentioned properties of  $\kappa$  hold for hinge loss, too.

The total loss function is convex, so a local minimum is also global. The optimum can be found with iterative optimization methods like L-BFGS (Zhu et al. 1997), stochastic gradient descent (Bottou 2003), or trust region Newton methods (Lin et al. 2007).

It is instructing to compare logistic regression to the well-known Naive Bayes model. As we already noted, the coordinates of  $\beta$  have the following interpretation: increasing coordinate  $x_j$  by 1 increases the estimation for the log-odds of class membership by  $\beta_j$ . Both for logistic regression and for Naive Bayes, increasing one of the variables by some constant increases the estimated log-odds of class membership by a constant amount. In fact, the maximum likelihood logistic regression model gives identical results to the Naive Bayes model if the Naive Bayes independence assumption is true. In this sense, logistic regression can be seen as a generalization of the Naive Bayes method that can robustly deal with non-independent features. Using a trivial example, if we accidentally add the same feature a second time to

our feature set, Naive Bayes double-counts the evidence, while logistic regression does not change its probability estimations.

A slightly more complex method for fitting the regression model is Maximum A Posteriori (MAP) estimation. In this case, we start from some simple prior probability distribution for the model parameter vector  $\beta$ , update the distribution using Bayes' Theorem after observing the training data, and choose the  $\beta$  with the maximum likelihood. Intuitively, we combine the suggestions of our training data with our prior knowledge of how  $\beta$  should look like. An analytically very tractable case is the Gaussian prior, where we assume that the coordinates of  $\beta$  are independent, normally distributed random variables with zero mean and fixed variance. It can be shown that with this prior, MAP estimation simply means that we have to add a quadratic regularization term  $\|\beta\|^2$  to the already described total loss function. This extra term punishes very large  $\beta_j$  weights, which helps avoid overfitting, as it prevents the model from becoming overconfident in the importance of specific features. Not only does the total loss function remain convex after adding the regularization term, it will be strictly convex, and thus have a unique optimum.

### 2.2.2 Entropy maximization

As we noted, the choice that we construct our probability function from a composition of a linear function and the logistic function seems a bit ad hoc. Below we claim that this exact functional form can be deduced from very general assumptions. In this subsection we do not even rely on the fact that the classification is binary.

We start without any assumptions about the structure of the probability function  $p(y|x)$ , except the most obvious ones: probabilities should be nonnegative, and sum to 1 over classes for each  $x$ . Next, we use the training corpus to assemble a set of linear constraints on the probability function. Each feature-class pair  $(j, y)$  will correspond to one such constraint. Before specifying the constraints, we define the Kronecker delta as  $\delta_{xy} = 1$  if  $x = y$ , and 0 otherwise. The constraints are



$$\sum_i p(y|x(i))x(i)_j = \sum_i \delta_{yy(i)}x(i)_j, \text{ for all } j, y.$$

(Summation over all training samples.)  $f_{u,j}(x, y) := \delta_{uy}x_j$  is called the indicator function defined by  $(u, j)$ . The constraint for a given  $(j, y)$  pair ensures that the expected value of the indicator function on the training corpus is equal to the expected value of  $x_j$  on the training corpus when class probabilities are provided by  $p(y|x)$  instead of being observed directly.

Let us use a toy example of a classification task where we have to classify fish into two classes ‘eel’ and ‘carp’ based on the values of features ‘length’ and ‘weight’. Let us assume that the expected values of the indicator function  $f_{eel,length}$  is 1.5 meters on the training data. (This is equivalent to saying that for the training data, the average length of eels multiplied by the ratio of eels is 1.5 meters.) We then constrain our probability function  $p$  so that for a randomly chosen fish  $x$ ,  $E(p(eel|x)x_{length})$  is 1.5 meters. Somewhat informally we can say that the constraint guarantees that the aggregate behavior of the feature is the same for the observed, actual ‘eel’ set and for the predicted ‘eel’ weighted set.

After we turned our set of features into a set of linear constraints on  $p(y|x)$ , we still have a vast class of feasible  $p(y|x)$  functions.<sup>4</sup> The next question is how to choose one from these. For this, we turn to the principle of maximum entropy. The principle of maximum entropy postulates that when we have to choose a probability distribution out of a class of probability distributions, we should choose the one with the maximal entropy. The justification for the principle is that this particular choice minimizes the amount of prior information built into the model, or as Ratnaparkhi phrased it, this model is “maximally noncommittal beyond meeting the observed evidence”. For the distribution defined by our probability function  $p(y|x)$ , entropy is calculated as

---

<sup>4</sup>We know that the set of feasible functions is non-empty, because the trivially overfitting probability function that mechanically gives back the training classification for the training data points obeys all our constraints.

$$H(p) = -E\left(\sum_y p(y|x) \log p(y|x)\right).$$

Here expectation is taken over the training corpus. It is easy to show that entropy is a concave function, thus the linear constraints and the concave objective function together define a concave optimization problem. It can be shown (Pietra et al. 1997) that this optimization problem and the optimization problem introduced in the previous subsection (maximum likelihood multinomial logistic regression estimation) are in a strong dual relationship with each other, which implies that the optimal  $p(y|x)$  is the same for these two seemingly quite different problems.<sup>5</sup> Note that unlike logistic regression, the maximum entropy approach did not start out with assuming a parametric form for  $p(y|x)$ . The specific parametric form was picked out from all the possible choices by the maximum entropy principle. This gives a further justification for the choice of the specific general linear model used for logistic regression.

## 2.3 Sequence labeling

In this section introduces three common approaches to the task of sequence labeling. Subsection 2.3.1 discusses Hidden Markov Models, Subsection 2.3.2 describes Maximum Entropy Markov Models, and Subsection 2.3.3 briefly introduces Conditional Random Fields.

The task of sequence labeling can be defined as follows: The input is a sequence of observations (we can assume that each observation is a feature vector), and the output is a label assigned to each of the observations. We assume that the ordering of the sequence carries information about the problem, so that we can incorporate information into our models about correlations between labels, or correlations between labels and features of other observations. Typically but not necessarily we want to model correlations between

---

<sup>5</sup>Technically,  $p(y|x)$  is a (not necessarily unique) optimum of the first function if it is a (not necessarily unique) optimum of the second.

labels and nearby information. For example we want to increase the probability of a token being a noun if the previous token turns out to be a determiner with high probability.

### 2.3.1 Hidden Markov Models

A standard class of models that can be used for supervised sequence labeling is Hidden Markov Models (HMMs) (Rabiner & Juang 1986), (MacKay 2002). Below we will introduce this class of models, and briefly describe the most important algorithms used when dealing with them.

Here we will introduce Hidden Markov Models as a special case of a model class we will call Hidden State Models (HSM). A HSM consists of two parts: a label sequence model (often called a language model) and an emission model. The label sequence model assigns probabilities to label sequences. Labels are called hidden states here, for reasons that will soon become clear. The emission model determines the probability of a given observation, conditioned on the fact that its label is already known. The label sequence model and the emission model together determine a joint distribution on (label, observation) sequences. This can be used to determine the label sequence with the largest likelihood, conditioned on the observation sequence.

The labels are called hidden states. This terminology reflects the fact that we look at the observations as if they were emitted by the labels. This approach is called the noisy channel paradigm, although the noisy channel is more a metaphor than an actual case of Shannon's original noisy channel framework (Shannon 2001), as noise means any sort of generative process here. To stay with the example of morphological disambiguation, the noisy channel paradigm looks at the sentence as if it was the result of the following strange process: some alien who talks in the language of morphological tag sequences utters such a tag sequence. The tags come through a noisy phone line, where due to noise the tags are distorted to become ordinary words (having the corresponding tags). The task of morphological disambiguation now becomes a decoding task: we need to find out the most probable utterance of the alien,

conditioned on the sentence observed, our probabilistic model of the alien language, and our probabilistic model of the line noise. The surprising power of this seemingly artificial approach comes from several sources, but one very important source of this power is that we have efficient algorithms to determine the most probable label sequence when the label sequence model obeys a Markov or ‘memorylessness’ property we will introduce now.

A Markov chain is a stochastic process generating random sequences (walks) over a finite set of states. It is determined by a probability matrix of transitions, where  $a_{ij}$  tells the probability that we move to state  $j$ , conditioned on the fact that we are at state  $i$ . (We can assume that the walk starts from state 0.) The crucial property of the Markov chain is that it is memoryless, in the sense that if we take the probability distribution of the sequences starting with a given prefix word, this distribution is identical to the probability distribution of the sequences starting with the last element of this prefix word. If we look at the left of this element as ‘past’, and the right of it as ‘future’, then the future is influenced by the past only through the ‘present’. A Markov chain assigns probabilities to state sequences, so it can be used as a label sequence model.

A Markov chain label sequence model together with some emission model is called a Hidden Markov Model (HMM). Thanks to the Markov property, there are efficient algorithms for answering many questions that one can ask about HMMs. First, it is trivial to calculate the probability of an observation sequence conditioned on a known hidden state sequence:

$$P(O_1, \dots, O_n | S_1, \dots, S_n) = P(O_1 | S_1) \dots P(O_n | S_n)$$

Second, it is trivial to calculate the unconditional probability of a state sequence:

$$P(S_1, \dots, S_n) = P(S_1) \prod_{i=2}^n P(S_i | S_{i-1})$$

Sometimes we would like to calculate the unconditional probability of an observation sequence. This is called the Evaluation Problem. For this, we need to sum over the set of all hidden state sequences:

$$P(O_1, \dots, O_n) = \sum_{S_1} \cdots \sum_{S_n} P(O_1, \dots, O_n; S_1, \dots, S_n) = \sum_{S_1} \cdots \sum_{S_n} \prod_{i=1}^n P(O_i | S_i) P(S_i | S_{i-1})$$

(By convention, define  $P(S_1 | S_0)$  to be  $P(S_1)$ .) The set of all hidden state sequences is exponentially large. Fortunately the sum can be calculated efficiently by the so-called forward algorithm. The algorithm is a straightforward application of dynamic programming. For each position  $i$  and for each state  $S$  it recursively calculates the probability that two conditions hold simultaneously: the first condition is that  $S_t = S$ . The second condition is that the first  $t$  observations are  $O_1, \dots, O_t$ .

Another useful task is to calculate the most probable hidden state sequence underlying a known observation sequence:

$$\arg \max_{(S_1, \dots, S_n)} P(O_1, \dots, O_n; S_1, \dots, S_n)$$

This is called the Decoding problem, and this is our task e.g. if we want to find the most probable morphological tag sequence for a given sentence within the noisy channel paradigm. For this, we can use an algorithm very similar to the forward algorithm, called the Viterbi algorithm. For each position  $i$  and each state  $S$ , the Viterbi algorithm recursively calculates the probability that two conditions hold simultaneously: the first condition is that  $S_t = S$ . The second condition is that  $S_1, \dots, S_t$  is the most probable state sequence for observations  $O_1, \dots, O_t$ , assuming we do not have information about later observations.

An easy, but still very useful task is supervised learning of HMM model parameters: given a hidden state sequence and corresponding observation sequence, find the model parameters that maximize the joint probability of the sequences. What makes this task particularly easy is that it decomposes into two tasks: the maximum likelihood state transition probabilities are equal to the empirical state transition probabilities, and the state-observation pairs can be used to train an emission model.<sup>6</sup> Note that if the Markov property or the conditional

---

<sup>6</sup>Depending on the structure of the observed states, the emission model can be quite complex. For example, in speech recognition applications a popular choice is a mixture of Gaussian distributions.

independence of emitted observations is violated, this can lead to badly biased maximum likelihood models.

We often need state sequence models where the conditional probability of the current state is a function of the last  $k$  states, not just the last single one. These are called  $k$ th-order Markov chains. Theoretically they are easy to work with, because if we treat the vector of the last  $k$  states as a single ‘big’ state, then the  $k$ th-order Markov chain becomes a regular Markov chain. In practice, we need to use a more compact representation of higher order Markov chains than this trivial one. HMMs of second order Markov chains are often called trigram HMMs, referring to the fact that their language model can be represented by a weighted set of trigrams.

### 2.3.2 Maximum Entropy Markov Models

HMMs are generative models in the sense that they can generate observation sequences. They model the joint probability of all variables of the model, in contrast to discriminative models that only aim to model the conditional probability of labels conditioned on the observations. Generativeness is a useful feature for several machine learning tasks, but discriminative models have advantages when solving the specific class of problems they are designed to solve, because they do not ‘waste’ modeling effort (model degrees of freedom) on modeling the distribution of input data.

Maximum Entropy Markov Models (MEMM) are designed to solve the same sequence labeling problem as HMMs, but they are discriminative models: they only attempt to model the behavior of the labels conditioned on the observations. MEMMs model the distribution of a label based on current observations and previous label:

$$P(S_1, \dots, S_n | O_1, \dots, O_n) = \prod_t P(S_t | S_{t-1}, O_t)$$

(Here each  $O_t$  is a feature vector, not just a single categorical variable.) A MEMM assumes that the distribution of  $P(S_t|S_{t-1}, O_t)$  is independent of position in the sequence, and models it with one separate maximum entropy model for each possible label  $S_{t-1}$  of the earlier state.

MEMMs have the important limitation that they implicitly assume that labels are marginally independent of later observations. This independence assumption is strongly violated in most sequence labeling tasks. For some of the results presented in this thesis (`hunchunk`, `hunner`), we will use a simplified version of Maximum Entropy Markov Modeling. This version differs in two important aspects from a standard MEMM. First, the maximum entropy model is allowed to depend on neighboring observations  $O_{t-k}, \dots, O_{t+k}$ , for some small constant  $k$ . Second, we do not let it depend on the previous label  $S_{t-1}$ . Rather, we model transition probabilities as conditionally independent from observations, and learn them directly from the training corpus. We will introduce this model in more detail in Subsection 5.3.3.

### 2.3.3 Conditional Random Fields

Conditional Random Fields (CRF) are a very general class of statistical modeling methods (Sutton & McCallum 2011). We only mention them here briefly, as no results in this thesis rely on CRF modeling.

For CRFs, the conditional dependencies between hidden variables can be represented by undirected graphs. For any given hidden variable, the following two conditional distributions are the same: 1. the distribution of the hidden variable conditioned on *neighboring* hidden variables and all observable variables. 2. the distribution of the hidden variable conditioned on *all other* hidden variables and all observable variables. Intuitively, other hidden variables can influence the variable only through its immediate neighbors.

When CRFs are used for some sequence labeling task, the graph is often (although not necessarily) a chain. In this special case called linear-chain CRF, the model can be seen as

an improved version of MEMMs, where the label of the present node need not be marginally independent of later observations, as is the case for MEMMs.

CRFs can model more complex dependencies than either MEMMs or our models. They achieve consistently higher scores on labeling tasks than MEMMs (Lafferty et al. 2001). They are theoretically more sound than our models. Linear-chain CRFs have efficient training and decoding algorithms. In spite of all these advantages, the results presented in the following chapters do not employ CRF methods. The reason for this is that our preliminary experiments suggested that the results we present can at best be only marginally improved by switching to linear-chain CRFs, while training time is at least one order of magnitude higher for CRFs than for our simpler models. As a result, this modeling approach is prohibitively slow for the very extensive feature sets we rely on. The accuracy of some of the systems presented in the thesis could likely be improved by using more careful feature selection combined with Conditional Random Field modeling.



## Chapter 3

# Morphological Analysis

In this chapter we introduce the task of morphological analysis. We describe the annotation formalism and morphological analysis framework built by our research group. We then present the Hungarian morphological resource that was built on these foundations. We start the thesis with these results because morphological analysis (MA) – and the `hunmorph` annotation formalism in particular – is an essential building block for all of our tools that is referred to in most chapters of the thesis.

This is the only chapter of the thesis where the author’s role in the work described is mostly supportive. Most of the results are by other members of the author’s research group. András Kornai and Péter Rebrus designed the `hunmorph` annotation formalism (Rebrus et al. 2012). Viktor Trón created the `hunlex/ocamorph` toolset (Trón et al. 2005), building on work by László Németh and others on the `hunmorph` analyzer (Németh 2002), (Németh et al. 2004). The Hungarian morphological description `morphdb.hu` was built by a large collaboration of authors (Trón et al. 2006). The role of the present author was mostly limited to building test environments for these tools and resources, and spotting lexicon problems by corpus-based semi-automatic methods.

### 3.1 The `hunmorph` annotation formalism

A morphological annotation formalism is a formal language that is suitable for encoding lexical information about word forms. It can be seen as a serialization of some abstract description of possible word structures. For a specific language, morphological annotation is a (many to many) mapping between word forms and elements of this formal language.

In this section, we describe the hunmorph annotation formalism and its instantiation for Hungarian.

In the following we briefly list the abbreviations used in this section. For more detail the reader should consult e.g. (Crystal 1997).

- PLUR: plural
- CAS: case
- ACC: accusative case, *ő*t ‘her’
- DAT: dative case, *Péternek* ‘to/for Péter’
- IMP: imperative mood, *Dolgozd fel!* ‘Process it!’
- SUBJUNC: subjunctive mood, *Szeretném, hogy feldolgozd.* ‘I’d like you to process it.’
- ANP: anaphoric possessive, *Péteré* ‘(something) belonging to Péter’
- FAM: familiar, *Péterék* ‘Péter’s group’

### 3.1.1 The general formalism

#### Hierarchical, asymmetrical

In the morphological theory behind the hunmorph formalism (Rebrus et al. 2012), the basic data structure is a labeled tree. Such a labeled tree corresponds to an equivalence class with regard to inflection. (In essence, such an equivalence class corresponds to a part of speech category. For example, the class of nouns has its own tree.) The vertices of the tree correspond to inflectional features. Importantly, hunmorph’s features are always binary. An annotation corresponds to a rooted subtree of the tree. This means that a daughter feature can be positive only if its parent feature is positive. The role of positive and negative features is asymmetrical: The string serialization of this rooted subtree is built from the positive features by bracketing. (The daughters of a given node are ordered in the full tree of features, to make

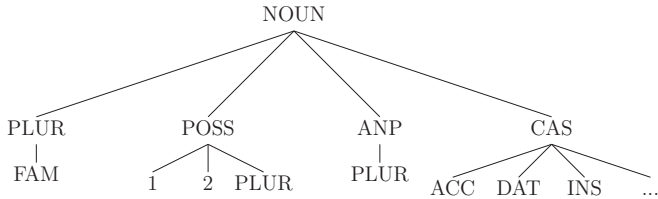


Figure 3.1: The signature of the graph originating from the root node *NOUN*

the serialization unique.) In Figure 3.1 we can see a part of the feature tree for Hungarian nouns. Some examples of annotation based on this are:

*kutya* ‘dog’

*kutya*/NOUN

*kutyának* ‘for/to the dog’

*kutya*/NOUN<CAS<DAT>>

*kutyáink* ‘our dogs’

*kutya*/NOUN<PLUR><POSS<1><PLUR>>

*kutyái* ‘those of the dog’ or ‘things belonging to the dog’

*kutya*/NOUN<ANP<PLUR>>

*kutyáikéi* ‘those of their dogs.ACC’

*kutya*/NOUN<PLUR><POSS<PLUR>><ANP<PLUR>><CAS<ACC>>

We can see that in this serialization the lemma is followed by a slash, then a specification of a tree (*NOUN* in the above cases), then a description of a subtree (for example <PLUR><POSS<1><PLUR>>). We will call the part before the slash the lemma, the part after the slash the inflectional tag. We will call the tree (the part between the slash and the first <) the POS-tag.

## Not segmentation-based

A natural idea when building a morphological formalism is to make it segmentation-based, that is, to incorporate a correspondence between segments of the word form and features of the morphological annotation. For example, *embereknek* ‘to people’ could be analyzed as *ember* (NOUN) *-ek* (PLUR) *-nek* (DAT). The authors of hunmorph do not believe that it is always feasible to have such a consistent correspondence between segments of a given word form and features of the annotation, so hunmorph does not attempt to provide such a correspondence. The authors give a list of examples that show that the segmentation-based approach is not feasible in general. We do not reproduce their arguments here (see (Rebrus et al. 2012)), but the main problem can be easily grasped through the following example: the Hungarian morph *-jaim* though corresponds to more than one morphological property (1st person, singular possessor, and plural possessed), these properties cannot be unambiguously associated with separate parts of the morph. The same problem can be present for suppletive (non-cognate inflected) forms, such as *benneteket* ‘you.PLUR.ACC’ or *gyere* ‘come.IMP’.

Another problem is grapheme-phoneme discrepancy. This is a less serious problem in itself, however it complicates implementation and usage. As an example, *hússzor* ‘twenty times’ is the inflected form of *hús* ‘twenty’. The only well-behaved and theoretically well-founded morph-based solution is to analyze it as *hús-szor*, but this is not a segmentation of the input character string.

It can be argued that this sort of morphological segmentation is not only infeasible, but not even useful in most applications. One exception where a segmentation-based approach actually has its benefits is speech recognition. There it is a justifiable simplification that leads to more tractable models. We mention that our annotation framework was easy to extend in order to provide segmentation information when such a need arose. The resulting segmentation is theoretically not well-founded, and behaves inconsistently for some irregular word classes, however when incorporated into a state-of-the-art morph-based speech recognition system, it was able to improve its error rate nevertheless (Mihajlik et al. 2007).

## Derivation

The above tree structure is intended to describe inflection, but is not directly suited to describe derivation. A derivational suffix can be seen as a relation between two part of speech classes, that is, a directed edge between the roots of two feature trees. (The two trees might be identical.) The POS category of the resulting word is the output category of the last derivational suffix, and the derived word can undergo further inflectional suffixing. Inflected forms, however, cannot be subjected to derivation. In the annotation formalism, the derivational suffix appears as an all-caps identifier in square brackets, and the name of its output category is placed after that. Examples:

<i>fax</i>	<b>fax/NOUN</b>	‘fax’
<i>faxol</i>	<b>fax/NOUN[ACT]/VERB</b>	‘to send a fax’
<i>faxolás</i>	<b>fax/NOUN[ACT]/VERB[GERUND]/NOUN</b>	‘faxing’
<i>faxolástól</i>	<b>fax/NOUN[ACT]/VERB[GERUND]/NOUN&lt;CAS&lt;ABL&gt;&gt;</b>	‘because of faxing’
<i>terjedő</i>	<b>terjed/VERB[IMPERF_PART]/ADJ</b>	‘growing’
<i>székestül</i>	<b>szék/NOUN[COM]/ADV</b>	‘together with the chair’
<i>sokszor</i>	<b>sok/NUM[MULTIPL-ITER]/ADV</b>	‘many times’

We call the part after the first slash the full morphological tag (as opposed to the part after the last slash, which we already introduced earlier as the inflectional tag).

## Compounding

In the hunmorph formalism, compounds are simply treated as the sum of their constituents, with two important restrictions. First: only the last constituent is allowed to be inflected. Second, it is allowed to have more than two constituents, but they are treated as a flat structure, with the last constituent determining the output category of the compound. For example, *(vad+hús)+evő* ‘eater of wild game meat’ and *vad+(hús+evő)* ‘wild eater of meat’

are not distinguished in the annotation. Preverb-verb constructions are treated as compound words.

<i>eladja</i>	el/PREV+ad/VERB<DEF>	‘she sells it’
<i>vérfarkasok</i>	vér/NOUN+farkas/NOUN<PLUR>	‘werewolves’
<i>sötétzöldje</i>	sötét/ADJ+zöld/ADJ<POSS>	‘its dark green’
<i>zúzottkő</i>	zúz/VERB [PERF_PART]/ADJ+kő"/NOUN	‘crushed rock’
<i>birsalmasajt</i>	birs/NOUN+alma/NOUN+sajt/NOUN	‘quince jelly’

### 3.1.2 The Hungarian instantiation

#### Part of speech categories

The valid POS categories are listed in Table 3.1. Inflectable categories are: ADJ, NOUN, NUM and VERB. The rest of the categories cannot be inflected.

Tag	POS category
ADJ	adjective
ADV	adverb
ART	article
CONJ	conjunction
DET	determiner
NOUN	noun
NUM	numeral
ONO	onomatopoeic
POSTP	postposition
PREP	preposition
PREV	preverb
PUNCT	punctuation
UTT-INT	utterance/interjection
VERB	verb

Table 3.1: Part of speech categories of hunmorph

### Inflection of nouns

We have already presented the feature tree for Hungarian nouns on Figure 3.1. The following restrictions apply to the combination of the features:

- the  $\pm CASE$  feature has to be continued by one of 16 cases,
- the features  $\pm 1$  and  $\pm 2$  exclude each other,
- if the  $\pm PLUR$  feature of  $\pm POSS$  is positive, then the  $\pm FAM$  feature cannot be positive,
- if the  $\pm PLUR$  and the  $\pm POSS$  feature are positive simultaneously, then the  $\pm FAM$  feature cannot be positive.

### Inflection of verbs

The feature tree for Hungarian verbs can be seen on Figure 3.1.2. (The tree has been cut into two parts for the sake of clarity.) In Hungarian IMP and SUBJUNC coincide at the level of morphology, so hunmorph uses the SUBJUNC-IMP code. The following restrictions apply to the combination of the features:

- only one of  $\pm SUBJUNC - IMP$  and  $\pm COND$  can be positive simultaneously,
- the feature  $\pm PAST$  can only be positive if both  $\pm SUBJUNC - IMP$  and  $\pm COND$  are negative,
- if the feature  $\pm OBJ$  is positive then its daughter feature has to be positive as well,
- the feature  $\pm INF$  can only combine with the feature  $\pm PERSON \pm PLUR$  and  $\pm MODAL$ .

Examples:

*lát* ‘he sees’

lát/VERB

*láttál* ‘you saw’

lát/VERB<PAST><PERS<2>>

*láthassátok* ‘that you may see it’

lát/VERB<MODAL><SUBJUNC-IMP><PERS<2>><PLUR><DEF>

### Inflection of pronouns

In Hungarian, a pronoun can substitute for any noun, adjective, numeral or adverb. The inflection of pronouns, where applicable, conforms to the restrictions described above. This enables us to avoid the use of ‘pronoun’ as a POS category, and to use instead those categories that the pronouns stand for. Personal pronouns are tagged as nouns in the hunmorph formalism because they take part in the same inflectional phenomena as nouns – although some of their paradigms are defective. They are distinguished by the *PERS* feature, however they are subject to the restriction that their *POSS* feature must be negative. Pronouns play the same syntactic role as the words they substitute for, so models that treat them similarly can generalize better. Sometimes, however, the information whether a word is a pronoun or not is still required, for example when writing grammars for noun phrases (Recski 2010). For these cases, we deduce this information from the stem by a small table lookup.

*ti* ‘you.PL’

ti/NOUN<PERS<2>><PLUR>

*titeket* ‘you.PL.ACC’

ti/NOUN<PERS<2>><PLUR><CAS<ACC>>

Possessive pronouns are personal pronouns with a possessed feature, thus they carry the *ANP* feature as well. Examples include:



*tiétek* ‘yours’

ti/NOUN<PERS<2>><PLUR><ANP>

*tieteknek* ‘to/for yours’

ti/NOUN<PERS<2>><PLUR><ANP><CAS<DAT>>

The anaphoric possessive can be repeated as shown in the following examples:

*enyémé* ‘that of my ...’

én/NOUN<PERS<1>><ANP<ANP>>

*tietekéi* ‘things belonging to something that is yours.PLUR’

ti/NOUN<PERS<2>><PLUR><ANP<PLUR><ANP<PLUR>>>

### 3.1.3 Comparison with other formalisms

The main candidate as an alternative for the hunmorph formalism is the MSD (Morphosyntactic Description) annotation scheme, originally proposed by (Erjavec & Monachini 1997). The big advantage of MSD is that a large concerted effort took place to implement it for many Eastern- and Central-European languages, and synchronize the annotation between these languages as much as possible. (We note that originally MSD was designed for Slavic languages only.) Like hunmorph, MSD does not deal with segmentation, and it serializes a sort of attribute-value structure. Still, there are several important differences that justify our research group’s effort to implement an alternative.

For MSD codes, the hierarchy of feature dependences is not made explicit, and is not consistently reflected in the serialization. MSD codes are fixed length, and features are identified with character positions. These design decisions make MSD codes harder to read than hunmorph codes, especially for less inflected words. Some comparisons:

<i>fiú</i>	Nc-sn-y---	NOUN
<i>faitokéival</i>	Nc-pi-yp2p	NOUN<PLUR><POSS<2><PLUR>><ANP<PLUR>><CAS<INS>>
<i>ad</i>	Vmip3s---n-----	VERB
<i>adtátok</i>	Vmis2p---y-----	VERB<PAST><PERS<2>><PLUR><DEF>

Moreover, MSD in its current form does not deal with derivation, and it is not immediately clear how such a generalization could be designed.

The Szeged Corpus, arguably the most important Hungarian language resource was annotated using MSD.<sup>1</sup> Thus, the need to convert between the two formalisms arose early. Originally this was carried out by using ad hoc scripts, with minor information loss in both directions. For a later version of the Szeged Corpus, a systematic harmonization effort took place, resulting in an annotated treebank that can be presented in either of the formats without information loss (Farkas et al. 2010).

### 3.2 Tools - hunlex, hunmorph

In this section we review the tools that do morphological analysis and related tasks. We roughly follow the presentation of (Trón et al. 2005). The author of the thesis was a co-author and minor contributor to this paper (working on semi-automatic corpus-based methods to detect bugs in the morphological descriptions). The software presented there are the work of László Németh (*hunmorph*), Viktor Trón (*ocamorph*, *hunlex*) and György Gyepesi (*jmorph*).

The task of a morphological analyzer is to provide all possible morphological analyses for a given word form. This task is closely related to a set of other word-level language processing tasks: Stemming, an important subtask of information retrieval (Halácsy 2006), is only interested in the stem of the word form (or possible stems, in case of ambiguity). The task of spell-checking is to decide whether a word has a valid morphological analysis. Note that this means that a spell-checking algorithm can be stopped after finding the first

---

<sup>1</sup>We will introduce the Szeged Corpus in Section 4.2.

valid analysis, which is typically not the case for morphological analysis or stemming. Spell-correction can be seen as providing the set of words that have valid analyses and, under some suitable distance measure, are closest to the input word. (This measure is usually an edit distance based on common misspelling patterns, for example missing accents, or letters close to each other on a keyboard.)

### 3.2.1 The runtime layer

The codebase we reused for the task of building a morphological analyzer has a long history in Unix-based text processing. Our **hunspell/hunmorph** codebase is a direct descendant of the **MySpell** spell-checking library by Kevin Hedricks, which itself is based on the earlier **ISpell** (Peterson 1980). The key operation supported by these algorithms is *affix stripping*. Affix rules are specified in a resource called the **aff** file. Such an affix rule consists of a list of conditions, a strip string, and an append string. For example, in the rule forming the plural of *body* the strip string would be *y*, the append string *ies*. The rules are reverse applied to complex input word forms. The result of the append and strip operations is then looked up in a base lexicon resource called the **dic** file. The two files together contain all the information needed to deal with a given language. We refer to this formalism and file format as **aff/dic** files.

Lexical entries are all associated with sets of affix flags, and affix flags in turn are associated to sets of affix rules. If the hypothesized base is found in the lexicon after the reverse application of the affix rule, the algorithm checks whether the flags of the lexical item contain the one that the affix rule is assigned to. In this sense, affix flags can be interpreted as lexical features. At the core of an affix stripping implementation such as **MySpell** or **hunmorph** is a fast indexing technique to check affixation conditions efficiently.

This simple table-lookup based mechanism does not scale well to languages with rich morphology. For instance, in Hungarian, due to productive combinations of derivational and

inflectional affixation, a single base can yield up to a million word forms. To solve this problem, **hunmorph** extends the affix stripping technique to a multistep method: after stripping an affix or cluster of affixes in step  $i$ , the resulting pseudo-stem can be further stripped in step  $i + 1$ . Affixes are associated with flags similarly to stems. As a useful byproduct, by cross-checking flags of prefixes on the suffix (as opposed to the stem only), simultaneous prefixation and suffixation can be made interdependent, enabling the correct handling of circumfixes like German participle *ge+t* or Hungarian superlative *leg+bb*.

Many languages use productive compounding extensively, therefore the treatment of this phenomenon is indispensable for wide coverage. **ISpell** allows a flag to specify whether a stem can appear in compound words. This limited solution leads to large classes of overgeneration, so **hunspell/hunmorph** generalizes it to deal with compound positions. A base or affix can be specified to occur only as leftmost, rightmost or middle constituent, and they can be marked to appear only in compounds. As an example of a problem that becomes tractable with this generalization, we mention the case of the German common noun: although it is capitalized in isolation, lowercase variants should be accepted when the noun is a compound constituent. In the **hunspell/hunmorph** solution, lowercasing itself becomes a prefix, with the compound flag enabled.

**MySpell** was not designed for morphological analysis, and one of our goals with the **hunspell/hunmorph** fork of **MySpell** was to make morphological analysis possible. This required eliminating from the codebase the assumption of halting after finding the first correct analysis. More importantly, it required functionality to emit output during analysis. Fortunately, this goal meshed nicely with the affix stripping algorithm. We extended the aff formalism to allow an output channel. For this channel, an optional strip-and-append operation can be given for each stripping rule and lexicon entry. Successively applying these operations creates an output string. When spell-checking, this functionality is not used. When stemming, only the output of lexicon entries is used. For full morphological analysis the full output is used to construct a morphological annotation.

We note that for a complex morphological annotation formalism like the `hunmorph` code, finding the emitted codes for each affix rule is not a trivial task, even when multistep affix stripping makes the set of affix rules more manageable. We will see a proper solution for this problem in the next subsection, where we introduce the `hunlex` morphological description formalism.

Two alternative implementations of `hunmorph` arose. One is `jmorph`, a fast Java implementation by György Gyepesi. (This was used as a stemmer in the Java-based runtime of our Hunglish bitext query system.) The other one is the OcaML implementation `ocamorph` by Viktor Trón.

### 3.2.2 The offline layer

For languages with complex morphology, the aff/dic formalism is far from being optimal as a description of morphology. Spell-checker resource developers for these languages typically use ad hoc precompilation tools to create the aff/dic resources from an ad hoc internal representation of the morphology of the language in question. The problem is further magnified when analysis requires lexicographic information.

The offline layer of our morphological toolkit seeks to remedy this by offering a high-level description language in which grammar developers can specify rule-based morphologies and lexicons. The `hunlex` morphological grammar precompiler can process these descriptions, generating aff/dic resources optimized for the runtime layer.

There are two kinds of rules employed by the grammar: morphosyntactically active rules and filter rules.

A typical example of a morphosyntactically active rule is the addition of an affix morph to a relative stem. These are grouped according to the morphosyntactic features they realize. Such a group cover the allomorphic variants of an affix morpheme. Figure 3.3 shows the ruleset for the Hungarian inessive case with its two allomorph rules *-ban* and *-ben*.

Filter rules describe morphophonological processes such as vowel shortening, and express redundancies between features and phonological patterns. An example for such a redundancy is the set of filter rules determining when a regular stem is lowering based on word form. These rules enable this feature to be provided only in the lexicon for irregular stems. The example on Figure 3.4 shows the rule describing that a word stem not marked in the lexicon as lowering can be treated as non-lowering.

Hungarian contains a significant number of loan words and proper nouns. The choice of affix allomorphs is regular given the pronunciation, but the pronunciation can not be deduced from the word form. To treat this class of cases, *hunlex* allows the specification of a pronunciation. (*Voltaire* /*volter*/, *Voltaire-rel* ‘with *Voltaire*’). The very general *hunlex* formalism also allows specification of transformation rules for grapheme to phoneme conversion, but currently the only places where this functionality is used are resolution of acronyms (*http*, *há-té-té-pé*), and the transcription of numbers.

Several parameters can be specified when we use *hunlex* to create an aff/dic resource. An important one is affix stripping recursion depth: *hunlex* can build a resource (although a possibly unreasonably large one) even when multistep affix stripping is not used during the runtime phase. More generally, the size of the aff/dic resource becomes smaller with higher allowed recursion depth, leading to a kind of time-space tradeoff for the runtime layer.

### 3.3 Resource built - morphdb.hu

Using the *hunlex* morphological description, we have built a wide-coverage morphological description for Hungarian called *morphdb.hu*. This resource can be compiled with the *hunlex* tool to an aff/dic resource for the *hunmorph* morphological analysis tool. The analyzer emits output conforming to the *hunmorph* morphological annotation formalism.

As we explained, the grammar contains filter rules that deduce features and stem variants successfully from the citation forms based on phonological and orthographic shape.

Thus, the dictionary entries only need to contain unpredictable irregularities, in the form of morphophonological and morphosyntactic features.

morphdb.hu consists of a wide-coverage lexicon, and a grammar of Hungarian inflectional and (productive) derivational morphology. The approximately 150k word lexicon is the result of a supervised merge of three pre-existing publicly available Hungarian language resources. The MagyarISpell dictionary is the spell-checking resource for **hunspell**. (For Hungarian spell-checking, MagyarISpell is still the de facto standard. The main cause of this fork is that the aff/dic created by **hunlex** automatically is larger than the manually built MagyarISpell aff/dic, and resource size is an important consideration for many spell-checking applications.) The MagyarISpell lexicon is the most up-to-date lexicon of present-day Hungarian, with more than 80 thousand lexicon entries (Németh 2002), (Németh et al. 2004). A second source of data was the Elekfi Dictionary of Hungarian inflections (Elekfi 1994). It contains about 66 thousand entries, classified into paradigm classes. The third source is a dictionary database edited by András Kornai (Kornai 1986) containing 78 thousand entries. Morphological information present in the three sources was transformed into morphophonological features used by the morphdb.hu grammar. This task required its own separate semi-automatic methods for each of the three sources.

After converting them into a common format, merging of the sources required the filtering-out of multiple occurrences, provided they were non-contradictory. Contradictory information was manually checked, revealing a significant number of errors and typos in all of the sources. The amount of overlap in the three sources turned out to be about a quarter of the overall size of the lexicon, with each of the three contributing at least 10 thousand unique entries.

The resulting resource was evaluated on two different datasets. First of these is the word set of the Szeged Corpus (Csendes et al. 2004), the language resource we used for many natural language processing tasks being introduced in the following chapters. As we already mentioned in the context of the hummorph morphological annotation formalism, The Szeged

Corpus contains morphological information on its tokens, in the form of so-called MSD codes. Having conversion scripts between MSD and `hunmorph` allowed us to automatically spot discrepancies between the codes output by `hunmorph/morphdb.hu` versus the Szeged Corpus annotations. After tuning our conversion scripts, the remaining discrepancies were resolved, mostly by increasing the coverage of `morphdb.hu`, but sometimes by tuning the Szeged Corpus annotations. The latest versions of `morphdb.hu` and Szeged Corpus are fully compatible (Farkas et al. 2010).

A second set of words to check `morphdb.hu` against was the Szószablya Hungarian Webcorpus built by a large-scale crawl of the Hungarian web. The creation of this resource will be detailed in the next chapter. The morphological analyzer was constantly checked against the set of most common word forms found in this corpus. In the end, after exclusion of the words recognized by an English spell-checker, the 40,000 most common word forms found in the corpus were checked against the morphological resource. In each case where `hunmorph` could not provide an analysis, manual inspection revealed that the word form is indeed either invalid or a rare proper noun.

### 3.4 Applications

A significant part of the material in the following chapters of the thesis can be considered as applications of the tools described in this chapter. Below we highlight some of the more interesting applications where the author of the thesis did not participate in the research.

An important application of the tools introduced above was the construction of a spell-checked and morphologically analyzed webcrawl of the Hungarian web (Németh et al. 2004) called the Hungarian Webcorpus. We postpone the introduction of this dataset until the next chapter on automatic morphological disambiguation, as the corpus is more useful in its morphologically disambiguated form (Kornai et al. 2006), and this later version is the one we intend to highlight.



Péter Halácsy (Halácsy 2006) compared several versions of a Hungarian information-retrieval system, all with the same off-the-shelf Lucene retrieval engine, but with several competing solutions for stemming. The results show that **hunmorph**-based lemmatization can significantly increase the precision of a retrieval system. In many cases, the system achieved higher precision than systems that used a state-of-the-art retrieval engine, but relied on some simpler form of rule-based stemming. It is worth pointing out that according to Halácsy’s results, automatic morphological disambiguation, the topic of the next chapter, does not increase the precision of a Hungarian information retrieval system.

We already mentioned a speech recognition application of the **hunlex/ocamorph** toolset (Mihajlik et al. 2007). For highly inflective languages it is a natural idea to build language models at the level of morphs instead of words as units. Usually this is achieved using an unsupervised morph-learner that deduces a useful set of morphs from a large corpus, and can segment unseen text into morphs based on this (Creutz et al. 2005). The advantage of this approach is that the learner is not bound to linguistically motivated segmentation patterns, and can directly optimize the complexity of the resulting language model, employing the Minimum Description Length principle. This approach is so successful that resource-based lemmatization and segmentation is rarely attempted when working on speech recognition tasks. Indeed, when Mihajlik et al. (Mihajlik et al. 2007) built **ocamorph/morphdb.hu** into their Hungarian speech recognition system, it was not competitive with the unsupervised morph-learning method. But they managed to combine the two approaches by using the Minimum Description Length principle to choose only from the set of possible segmentations suggested by **ocamorph**. The resulting system slightly outperformed the unsupervised system and significantly outperformed the word-based system.

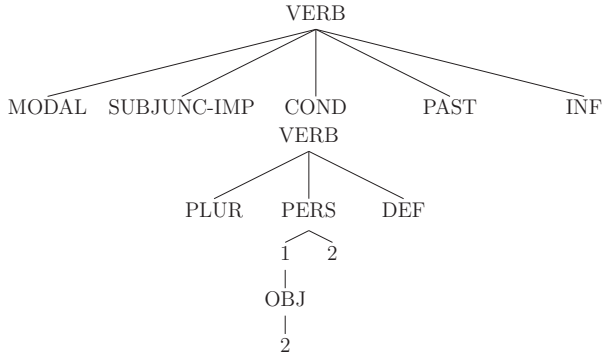


Figure 3.2: The signature of the graphs originating from the root node *VERB*

```

CAS_INE
  IF: analytic lengthened cas_ine
  TAG: <CAS<INE>>

, +ban IF: back
, +ben IF: front

;

```

Figure 3.3: The rules separated by commas refer to the allomorphs of the suffix, while the whole rule (CAS\_INE) refers to the inessive case morpheme.

```

NOM_LOWERING_FILTER

  FREE: false
  FILTER: low non_low
  OUT: NOM_KEEP_ALL_FEATURES
  OUT: NOM_ACC_FILTER

,      OUT: non_low
;

```

Figure 3.4: The lowering filter: example of associating a default feature.

## Chapter 4

# Morphological Disambiguation

In any morphologically complex language, morphological analysis will often return more than one possible analysis for a given word. For many automatic text processing applications, we need to decide which of these alternatives is the correct one. This morphological disambiguation task is closely related to, but not identical with, part of speech (POS) tagging, a term we reserve for finding the major parts of speech (noun, verb, etc).

The chapter starts with the description of the task of morphological disambiguation. We then proceed to highlight some complications presented by morphologically rich languages such as Hungarian, and present an analysis of the hardness of the task. Next we describe, evaluate, and compare those systems that we designed to solve the task of morphological disambiguation for Hungarian. We then proceed to present our morphologically disambiguated Hungarian webcorpus and frequency dictionary. Finally we highlight some of the applications made possible by the availability of these datasets.

The main results of the chapter originally appeared in (Halácsy, Kornai & Varga 2005) (in Hungarian) and (Kornai et al. 2006). The author of the thesis was an equal collaborator with Péter Halácsy and András Kornai in the design and creation of the morphological disambiguator systems presented (Halácsy, Kornai & Varga 2005). The creation of the morphologically disambiguated Szószablya webcorpus and frequency dictionary was the result of a larger collaboration (Kornai et al. 2006). The author contributed software tools to an ongoing project to build webcorpora and frequency dictionaries for medium and small density languages. This work is documented in (Halácsy et al. 2008) and (Zséder et al. 2012). The subsection on a psycholinguistics application is the result of a collaboration with Csaba

Pléh and co-workers (Pléh et al. 2011), where the author’s contribution was the construction of the entropy models, and the statistical analysis of the empirical data.

## 4.1 Background

Morphological Analysis (MA) is a central task in processing languages with rich morphology such as Hungarian: from spell-checking to machine translation there is hardly any practical application that does not require some form of MA. Even with a perfect MA algorithm that recognizes every word and never makes mistakes, one still has to deal with the fact that several word forms in Hungarian are ambiguous and the correct analysis can only be chosen on the basis of context. As an example, the Hungarian word *ment* can be analyzed as either *megy*/VERB<PAST> ‘*went*’ or *ment*/VERB ‘*saves*’, and even has a third, archaic analysis *ment*/ADJ ‘*free of*’.

In this chapter we will focus on the task of morphological tagging and disambiguation. Morphological disambiguation is the task of assigning a unique morphological analysis to each token of a text.

A full morphological tag contains both POS information and morphological annotation: in highly inflecting languages the latter can lead to tagsets of high cardinality (Tufiş et al. 2000). Hungarian is particularly challenging in this regard – with the number of individual tags in the thousands.<sup>1</sup> As a result, the ratio of tokens that are not seen during training (unseen) can be as much as four times higher than for English corpora of comparable size. Moreover, the number of ambiguous tokens is high (reaching 50% in the Szeged Corpus according to (Csendes et al. 2004)).

Fortunately, morphological disambiguation is easier to automate than its semantic analogue, word sense disambiguation (Preiss & Stevenson 2004). As we will see in this chapter,

---

<sup>1</sup>The number of inflectional tags in our largest corpus (see Section 4.8.) is 2004. The number of full morphological tags is above 20000, which is not surprising if we consider that derivations can be combined almost without limits.

relatively simple statistical modeling of the local context of the word is sufficient for disambiguation with a precision acceptable for many applications.

We approach the problem through the subtask of assigning inflectional tags, and this subtask will be the major focus of the chapter. All of the results will be presented for Hungarian, a morphologically complex language, but our methodology and our tools are language independent. (Later, related work by our co-authors led to state-of-the art results for English (Halácsy et al. 2007), Swedish (Megyesi 2009) and other languages.)

## 4.2 Resource used - the Szeged Corpus

The Szeged Corpus is the largest manually annotated natural language corpus of Hungarian (Csendes et al. 2004). It was created as a joint effort of the University of Szeged Department of Informatics, MorphoLogic Ltd. Budapest, and the Research Institute for Linguistics at the Hungarian Academy of Sciences. It consists of 1.2 million tokens from 145 thousand different word forms, and an additional 225 thousand punctuation marks. Its material was carefully selected to represent several different genres in a balanced fashion. It consists of the following subcorpora:

- fiction: two Hungarian novels and the Hungarian translation of Orwell's 1984. ~187k tokens.
- short essays of 14-16-year-old students. ~223k tokens.
- newspaper articles: excerpts from three daily and one weekly paper. ~187k tokens.
- computer-related texts: excerpts from a Windows 2001 manual and some issues of the ComputerWorld, Számítástechnika magazines. ~182k tokens.
- law: excerpts from legal texts on economic enterprises and authors' rights. ~222k tokens.
- short business news: from the archive of the Hungarian News Agency. ~188k tokens.

The text was first run through the `Humor` syntactic analyzer (Prószték & Tihanyi 1996) that produces MSD codes. Ambiguities were then manually resolved by trained linguists, incorporating over 124 person-months of manual work.

Our toolchain is centered around the `hunmorph` annotation formalism, so we created a version of the corpus where MSD codes were automatically converted to `hunmorph` codes.<sup>2</sup> The conversion step is not always straightforward, since the two systems differ in their use of inflectional features (e.g. in the case of marginal case suffixes and in the treatment of the familiar plural). The resulting tagset (both MSD and `hunmorph`) is very large compared to morphologically less complex languages.<sup>3</sup> In all we converted 1001 MSD codes to 744 `hunmorph` tags, which may appear to have simplified our task; however, the `hunmorph` tag and stem allow for a 100% reconstruction of the MSD tag, which means that the merging of tags does not cause any information loss. In other words: Using a static table, a `hunmorph` tagging of given precision can be converted to an MSD tagging with equal or higher precision.

Sentences containing MSD-tags of the residual main categories (X, Z, O) were omitted from the corpus. Although `hunmorph` recognizes a number of ‘X’ items, and `hunspell`, which uses the same list of stems, can correct many items with label ‘Z’ (typos), Szeged Corpus does not provide corrected codes (ground truth) for these elements, which makes the evaluation of our system’s output impossible on such data. With regard to the open class of tags of main category label ‘O’, our experience showed that these items are not always distinguishable even by human annotators. In the end we kept 70,084 sentences of the 82,098 originally present in the corpus.

Named entities are treated as a single unit in the Szeged Corpus, with a single MSD code. Although the recognition of named entities (NER) would constitute a separate task, we kept tokens containing space, which make up 1.37% of the corpus. These increase the number of OOV words in our experiments.

---

<sup>2</sup>See Subsection 3.1.3 for a brief comparison of the MSD and `hunmorph` annotation formalisms.

<sup>3</sup>The Penn Treebank tagset (Mitchell et al. 1994), the most commonly used tagset for English has 36 tags.

It is worth noting that the Szeged Corpus has since been annotated with syntactic information at the sentence level. This improved version is called The Szeged Treebank (Csendes et al. 2005), and we will discuss it in more detail in Chapter 6 when we introduce the task of noun phrase chunking.

### 4.3 Related work

In this section we introduce earlier approaches to morphological disambiguation, and use this opportunity to contrast our approach with these. Section 4.5 will provide a more detailed description of our models.

In our approach, lemmatization and full morphological disambiguation is postponed to a later rule-based postprocessing step after the inflectional tag was determined, as in (Erjavec & Džeroski 2004). This differs from the method of (Hakkani-Tür et al. 2000), where all syntactically relevant features (including the stem or lemma) of word forms are determined in one pass.

In Hungarian there is rarely tag ambiguity at the derivational level, but lexicalization is an important issue. The choice of stem depends so heavily on the type of linguistic information that later processing will need that it cannot be resolved in full generality at the morphosyntactic level.

To the best of our knowledge, before our work the highest precision rates for the task of Hungarian inflectional tagging (more concretely and precisely MSD-tagging) were reported by (Oravecz & Dienes 2002), with an accuracy of 98.11% on their corpus. They modified the HMM-based TnT tagging system (Brants 2000) for the task: they used the Humor (Prósztéký & Tihanyi 1996) Hungarian morphological analyzer to tag the tokens not occurring in their training corpus, and to restrict the set of possible tags to those that are compatible with the output of the MA. (This strategy was already employed by (Hakkani-Tür et al. 2000, Hajić et al. 2001) for other languages.)

Our system improves on this solution mainly by giving good probability estimates for the tags of the words that are recognized by neither the training corpus nor the morphological analyzer. For morphological analysis we used the `hunmorph` system with the `morphdb.hu` language resource.

Our precision rates of 97.91% on 1984, 98.38% on the `wholenews` section<sup>4</sup> and 98.17% on the Szeged Corpus do not indicate a substantial improvement compared to the numbers reported by (Oravecz & Dienes 2002). However, we still believe our system to be of better practical use, because our algorithm is robust in handling coverage errors of MAs not adjusted to the corpus at hand, thus enabling us to process larger and more heterogenous corpora such as that created from the dynamically expanding Hungarian web. We note that after the completion of this project we started and completed the task of making the Szeged Corpus fully covered by our morphological analyzer (Farkas et al. 2010). This later work did not taint the results published here that are based on a version of the MA not specifically tuned to the Corpus.

#### 4.4 The hardness of the tagging task

The difficulty of the morphological disambiguation task is usually measured by the ratio of ambiguous word forms (Csendes et al. 2004) or the average number of possible analyses for a token (Tufiş et al. 2000). If the lexicon offers alternative analyses, the token is taken as ambiguous irrespective of the probability of the alternatives. These numbers, however, can be quite misleading. If an external resource is used in the form of a morphological analyzer (MA), this will almost always over-generate, yielding false ambiguity. But even if the MA is tight, a considerable proportion of ambiguous tokens will come from legitimate but rare analyses of frequent types (Church 1988). For example the word *nem*, can mean both ‘not’ and ‘gender’ in Hungarian, so both `ADV` and `NOUN` are valid analyses, but the adverbial reading

---

<sup>4</sup>The 280 thousand token corpus used by Oravecz et al. (Oravecz & Dienes 2002) is most comparable to the 350 thousand token subcorpus of the Szeged Corpus we call `wholenews` that consists of the news and newsm1 (business short news) subcorpora.



is about five orders of magnitude more frequent than the noun reading, (12596 vs. 4 tokens in the 1m word manually annotated Szeged Corpus (Csendes et al. 2004)).

In what follows, a distinction must be made between those items that are not found in the training corpus (these we have called *unseen* tokens) and those that are not known to the MA – we call these out of vocabulary (OOV). As we shall see, the key to the best tagging architecture we found was to follow different strategies in the lemmatization and morphological disambiguation of OOV and known (in-vocabulary) tokens.

A more relevant measure for the difficulty of the disambiguation task is the average amount of information necessary for disambiguating a word form. If a word  $w$  has tag  $T_i$  with probability  $P(T_i|w)$  (which can be approximated with  $C(T_i, w)/C(w)$ ,  $C$  being the number of occurrences in a tagged corpus), then the tag-entropy of this word is  $H(w) = -\sum_i P(T_i|w) \log P(T_i|w)$  and the difficulty of the entire tagging task is the weighted average of these entropies based on word frequency  $\sum_w P(w)H(w)$ . For the Szeged Corpus this is approximately 0.1 bit/word (the exact value depends on the chosen tagset), far lower than what one might predict from the ratio of ambiguous words: with half the words in the corpus having two equally probable analyses, entropy could be as high as 0.5 bit/word.

In practice, no morphological analyzer is perfect and both word frequency and tag-entropy can only be estimated. Of particular interest are methods that make these estimates without analysis, directly from the corpus, as these correspond to algorithms that use only learning and no MA. Such methods provide quite good solutions to the tagging task in themselves: e.g. assigning the most frequently observed tag to each word and the most frequent tag ‘NOUN’ to unseen words will accomplish a precision rate of 92% on the Szeged Corpus (90% train, 10% test, unshuffled tenfold cross-validation). (Oravecz & Dienes 2002) use the same algorithm as baseline but report a precision of only 81.2%. The difference is due to the fact that our training and test corpora are an order of magnitude larger, which reduces the ratio of unseen words to 6.75% from the 17.13% found in their corpus.

(Oravecz & Dienes 2002) already notes that the richness of Hungarian morphology causes the ratio of unseen words to be higher in a corpus of Hungarian than in one of English (for 270,830 tokens they report 17.13% and 4.5% respectively). Since the ratio of unseen words strongly influence the efficiency of methods more complex than the baseline, there are three methods we can choose from: (A) increasing the size of the training data to decrease this ratio, (B) linking unseen words to seen words, or (C) improve our heuristics for unseen words, e.g. by using MA. Our results will give some evidence on the relative influence of these methods on precision.

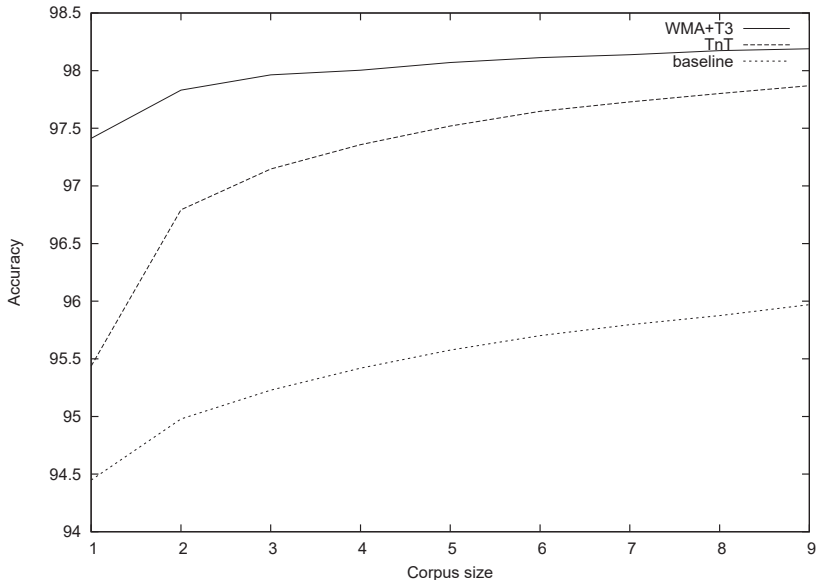


Figure 4.1: Learning curves of various algorithms on a mixed corpus

A good example of method (B) is the following modification of the baseline algorithm (a similar one was suggested by (Horváth et al. 1999)):

- (i) If  $w$  is in the training corpus, it is assigned tag  $T = \operatorname{argmax} P(T_i | w)$ , otherwise

- (ii) if the MA recognizes the word and provides a single tag, than that tag is assigned,
- (iii) if the MA recognizes the word as ambiguous, then we choose from all possible tags  $T_{w,i}$  the one most frequent in the training corpus, in all remaining cases
- (iv) the word is assigned the tag NOUN

This algorithm accomplishes precision rates of 95.40% and 95.84% on Szeged Corpus and *1984* respectively, a performance that is comparable to that of transformation-based learning systems ((Horváth et al. 1999), (Kuba et al. 2003), (Kuba et al. 2005)), but is inferior to the 98.11% achieved using trigram HMMs ((Oravecz & Dienes 2002)) Since the method achieves high precision for words already seen and the ratio of unseen words decreases with the increase in corpus size, overall precision can be increased along with the size of the data, as seen in Figure 4.1. (The unit on the  $x$  axis is 10% of total corpus size. A random 10% of the corpus is held back for evaluation.)

A note about the corpus used for plotting Figure 4.1: the Szeged Corpus consists of several sections which differ greatly from one another in both genre and difficulty. To prevent systematic distortions caused by this, for these measurements we randomly shuffled the sentences of the corpus before splitting it into training and test sections. Precision rates measured on this shuffled corpus can not be compared to with our other numbers measured using unshuffled tenfold cross-validation, since random shuffling greatly reduces the ratio of unseen words in the test corpus. In this sense, our other published numbers are for a harder task. We consider this task more realistic than the shuffled version, because in real-life applications the input rarely comes from the exact same probability distribution as the training data.

Figure 4.1 also demonstrates the effects of the morphological analyzer. The HMM-based TnT that does not use any MA outperforms the simple MA-based model because it can make use of the word’s context. If the trigram-model is extended so that it can fall back to MA output in the case of unseen words (the solution employed by (Oravecz & Dienes 2002)),

precision will increase substantially. However, the figure also shows that the positive effect of the MA will decrease with the increasing corpus size (and thus decreasing ratio of unseen words).

As can be expected, the most prominent source of errors for morphological disambiguators are words that are neither present in the training corpus, nor recognized by the MA (out of vocabulary, OOV). These words constitute 2% of the test corpus. For a given corpus, OOV can be arbitrarily decreased or even completely eliminated by expanding the stem-list of the MA. But in the long run, dynamically growing corpora such as the Hungarian web will exhibit OOV-s above 2%, since the vocabulary is constantly expanding, especially due to new proper names. From this perspective, the approach of building MA from the corpus before splitting it into train and test sections will simply eliminate the question of OOVs from the experiments and therefore render the algorithms' potential to achieve similar performance on new, unseen corpora questionable.

## 4.5 Our algorithms

Method (B) discussed in the previous section uses morphological analysis to associate the input with words already seen in the training data. Our architecture, however, is closer to method (C): improving the heuristics for unseen words. This architecture goes beyond the ambiguity classes obtained from the MA and provides an explicit estimate for tag-probabilities.

Maximum entropy models were first used for POS-tagging by (Ratnaparkhi 1996). From now we shall regard sentences as word sequences  $(w_1, \dots, w_n)$ , for which a sequence of tags  $(t_1, \dots, t_n)$  is available at the time of training.

Our first maximum entropy model is not a real sequential tagging model: it consists of learning a probability function  $p(t_i | w_{i-k}, \dots, w_{i+k})$  that assigns probabilities to tags based on local context  $w_{i-k}, \dots, w_{i+k}$ . The assumption behind the model is that  $p$  is independent of token position  $i$ .

Our task is to turn a local context into a feature vector. Our model works with  $k = 1$ , only looking into the previous, current, and next token when assigning a probability to some tag for the current token. Here is the full list of features for our model:

1. the word form in lowercase
2. the ambiguity class obtained from the MA output
3. whether the token contains numbers or non-alphabetic characters
4. whether the token is all-uppercase or capitalized
5. for words longer than 5 characters, the last 2, 3 and 4 characters
6. lowercase form of the previous word in the sentence, if any
7. lowercase form of the next word in the sentence, if any

Some of these are individual binary features (e.g. whether or not the word is capitalized), but most of them are feature groups. As we already noted in Section 2.1, a feature group like *the last three characters of the word* is to be interpreted as a very large set of binary features, one for each possible value for the three-character suffix. The training algorithm only assigns weight to the features observed in the training corpus, that is, having value 1 on at least one data point. Thus, the number of features the training algorithm has to deal with stays below some linear bound of the training corpus size.

The best way to convert the MA output to features is not immediately clear. The best results were obtained by representing the set of candidates provided by the MA (the so-called ambiguity class) with a single feature group. Suffix and form features are used primarily to handle OOV words: in case a word is neither recognized by the MA, nor present in the training corpus, the model makes use only of features describing the suffixes and neighboring word forms.

The maxent model does not decide on any tag for a given token – it simply gives the probability of each possible tag. By combining the maxent model and the MA we can build a context-sensitive weighted morphological analyzer (WMA), which assigns to each word a probability distribution of tags based on context in the following way:

1. If the MA recognizes the word, we allow tags in its output and normalize their probabilities from the maxent model to 1. (In particular, if MA knows one analysis only, we assign probability 1 to that.)
2. In case of an OOV word form we allow the three word forms that are most probable according to the maxent model and normalize their probabilities to 1. (The number three has been chosen arbitrarily to increase computational speed. Since the total probability of the candidates discarded is typically very low, the precision of the system will not increase by choosing larger values.)

A simple disambiguator based on the context-sensitive WMA can choose the most probable tag given for each word. Below we will call this disambiguator **MA+ME**. Note that the context-sensitive WMA can easily be turned into a context-free one by omitting features representing the surface forms of neighboring words.

This simple **MA+ME** model does not directly utilize information available about the tag probabilities of neighboring words. (Although it does utilize context information, in the form of features of neighboring words.) To exploit this sort of contextual information, we created two other models.

The first, called **WMA+T3** is a variation of trigram HMM. It uses the tag probabilities provided by the context-free WMA together with a tag trigram language model to assign likelihoods to tag sequences. This system is quite similar in design to the one described by (Oravecz & Dienes 2002). The main difference is that instead of relying on TnT’s suffix tree algorithm, we use the maxent method to estimate tag probabilities for unseen words.

The Viterbi algorithm can be used to find the maximum likelihood tag sequence under this model.

Our last model, called **TMM** for TnT+MA+ME gives another, arguably more ad hoc solution to the problem of exploiting tag context dependence. It relies on the context-sensitive WMA, trained with two more features in addition to the ones we already discussed: the tags of the previous and next word. More formally, this means two binary features for each possible tag, e.g. `previous.NOUN` and `next.VERB<PAST>`. In the training phase these are provided as ground truth, and in the tagging phase these are predicted by the TnT tagger.

## 4.6 Evaluation

For evaluation, we used tenfold cross-validation. We did no random shuffling of the sentences of the corpora, using consecutive blocks as test data. (Note that this means that the 10% test corpora used during cross-validation is often quite different in genre and content from the corresponding 90%, so the task is significantly harder than random tenfold cross-validation.)

On Table 4.1 we report precision rates for several subcorpora of the Szeged Corpus. After size of subcorpus and percentage of OOVs in subcorpus, the table shows scores for all six algorithms we described above. They are: Baseline (most probable tag for seen words, NOUN for unseen words), BMA (modified version of baseline that makes use of MA), TnT (Brandt’s tagger, not using MA), MA+ME (context-dependent maximum entropy model augmented with MA features), WMA+T3 (the context-free version of the previous model combined with a tag trigram language model), TMM (context-dependent maximum entropy model augmented with MA and TnT features). We can see that the ranking of the systems is completely independent from the chosen subcorpus.

The performance of the purely statistical TMM system is superior to all rule-learning systems that we know of: (Kuba et al. 2005) reports precision rates of 96.52% and 98.26% for the entire Szeged Corpus and its news section respectively. As (Kuba et al. 2003) notes,

Subcorpus	size	OOV	Baseline	BMA	TnT	MA+ME	WMA+T3	TMM
Literature	209785	5.79	86.20	95.46	96.02	97.37	97.63	97.83
Children's	290167	1.62	90.17	96.34	96.97	97.73	97.80	98.01
Press	355311	9.98	82.68	94.36	97.32	97.93	98.14	98.38
IT	157969	8.43	86.06	94.44	97.02	97.53	97.91	98.11
Law	147766	4.97	91.41	96.89	98.44	98.76	98.96	99.04
Total	1161016	5.64	89.70	95.40	97.42	97.72	97.93	98.17

Table 4.1: Precision rates of our algorithms on various subcorpora.

simpler systems use rules that are easily manageable, understandable and manually extendable. These advantages are lost with the increase of precision, however, due to the sudden increase in the number and complexity of rules necessary for performance comparable to statistical systems. The process of using the corpus to build ideal MAs that recognize every word is methodologically problematic, as this makes it impossible to run the system on new, unseen text. For example, (Horváth et al. 1999) achieves a precision rate of 98.03% on *1984* on which our system performs 97.91% only. However, once we exchange our independent MA with the morphological dictionary built from the corpus, we achieve 98.50% rate of precision under the same conditions.

Our results show that purely statistical systems can be effectively combined with rule-based MA. For Hungarian, this was first shown by (Oravecz & Dienes 2002). Our system's advantage over theirs is its robust treatment of OOV words. Our results are not fully comparable, since the two systems have been evaluated on different corpora (though similar in size and genre). The precision rate of 98.17% for the entire corpus is remarkable not only because of the treatment of OOV words, but also because it is a result of unshuffled cross-validation on a heterogeneous corpus that is built up from documents of various genres. Our goal with the creation of these systems was to enable the morphological analysis of corpora of greater variability, such as those derived from the dynamically growing Hungarian web.



## 4.7 Later work

After the publication of these results, co-author Péter Halácsy created another high-precision morphological disambiguator system (Halácsy et al. 2007). This is the component that solves the task in our current text processing framework. The underlying tagger is called **hunpos**, and the morphological disambiguator itself is called **hundisambig**.

**hunpos** does not use maximum entropy learning. Rather, it follows the architecture of Brandt’s TnT (tag n-gram HMM, with the suffix guessing emission model for unseen words), but with the ability to incorporate MA information, using the output of the MA to constrain suffix guessing when meeting unseen words. The precision of this new algorithm is roughly equal to the algorithm we described above: on the Szeged Corpus it achieves a 98.24% precision versus the 98.17% precision we reported. It has two important advantages, though: first, training is faster by two orders of magnitude, since building a suffix guessing model is much less computationally expensive than building a maximum entropy model. Second, it has much simpler architecture and a fewer number of software dependencies. As an improved, open source reimplementation of TnT, it is frequently used for the task of POS-tagging and disambiguation (e.g. (Megyesi 2009)), and is one of the standard building blocks (e.g. (Fuschetto et al. 2009)) and baselines (e.g. (Silfverberg & Lindén 2011)) when creating new POS-tagging systems and evaluating their precision.

For Hungarian, the Szeged Natural Language Processing Group created an open source morphological tagger by adapting the Stanford Tagger (Toutanova et al. 2003). The adapted tagger is part of the **magyarlanc** framework (Zsibrita et al. 2009) that builds on the UIMA (Ferrucci & Lally 2004) framework. We are not aware of any published performance measures.

## 4.8 Resource built - the Szószablya corpus and lemmatized frequency dictionary

Frequency dictionaries play an important role both in psycholinguistic experiment design and in language technology. The section describes our freely available morphologically disambiguated webcorpus and frequency dictionary of Hungarian that is being used for both purposes, and the language-independent techniques used for creating it.

The corpus gathered is based on 18m pages crawled. Its best quality stratum (see below) consists of 589m words harvested from 1.22m pages. As a comparison, the Hungarian National Corpus ([http://corpus.nytud.hu/mnsz/index\\_eng.html](http://corpus.nytud.hu/mnsz/index_eng.html)) (Váradı 2002) has 187.6m words, the manually annotated Szeged Corpus (Csendes et al. 2004) has 1.2m words.

### 4.8.1 The data processing pipeline

**Raw data, preprocessing** The raw dataset comes from crawling the top-level domain, e.g. `.hu`, `.cz`, `.hr`, `.pl` etc. Pages that contain no usable text are filtered out, and all text is converted to a uniform character encoding.<sup>5</sup>

Identical texts are dropped by checksum comparison of page bodies (a method that can handle near-identical pages, usually automatically generated, which differ only in their headers, datelines, menus, etc.) For normalization we use `hunnorm`, which performs HTML stripping and character conversion to produce uniform text files from web pages. It uses a `flex` pipeline and relies on existing open source code for encoding conversion and file type determination.

**Sentence segmentation, tokenization** Next we detect sentence boundaries and tokens by the `huntoken` module, a rule based tokenizer written in `flex` which is similar in concept and design to the rule system described by (Mikheev 2002). It employs 25 regular-expression rules, and relies on an approximately 150-word list of common abbreviations. Evaluated

---

<sup>5</sup>In the case of the original Szószablya Corpus we describe here, this uniform encoding is ISO Latin-2. See Subsection 4.8.3 for the description of our latest pipeline, which is Unicode-based.

against the Szeged Corpus, `huntoken`’s sentence boundaries are incorrect in 1064 cases out of the 86094 sentences, yielding an error rate of 1.3% which is significantly better than the simple regex `[.!?]` baseline of 6083 (7.0%).

**Stratification** The `hunspell` spell checker is used to stratify pages by recognition error rates. For each page we measure the proportion of unrecognized (either incorrectly spelled or out of the vocabulary of the spell-checker) words. To filter out non-Hungarian (non-Czech, non-Croatian, non-Polish, etc.) documents, the threshold is set at 40%. If we lower the threshold to 8%, we also filter out *flat* native texts that employ Latin (7-bit) characters to denote their accented (8 bit) variants (these are still quite common due to the ubiquity of US keyboards). Finally, below the 4% threshold, webpages typically contain fewer typos than average printed documents, making the results comparable to older frequency counts based on traditional (printed) materials. Table 4.2 shows the effect of stratification on the size of the resulting corpus.

<i>t</i> (%)	100	40	8	4
pages (m)	3.493	3.125	1.918	1.221
tokens (m)	1486	1310	928	589
types (m)	19.1	15.4	10.9	7.2
hapaxes (m)	11.5	8.9	6.3	4.2

Table 4.2: Stratified corpus size

**Lemmatization** To turn a given stratum of the corpus into a frequency dictionary, one needs to collect the word forms into lemmas. This is done at the sentence level, as this is the level where `hunpos` can utilize context to resolve inflectional ambiguities. Analyses are computed by our `ocamorph` morphological analyzer. The choice between alternative morphological analyses is resolved using the output of the `hundisambig` morphological disambiguator. (`hundisambig` is just a thin wrapper around `hunpos` that delegates the work of finding inflectional tags to `hunpos`.) When the word has at least one valid analysis according to `ocamorph`, `hundisambig` chooses from the tags compatible with the output of `ocamorph`.

When there are several analyses that match the output of the tagger, **hundisambig** chooses the one with the least number of identified morphemes. (The rare case of ties is broken lexicographically.) The result of this is that during lemmatization inflectional tags are removed but derivational affixes are removed only if the derived word is not present in the dictionary. This is a good heuristic to deal with lexicalization.<sup>6</sup> Words outside the vocabulary of the MA are not lemmatized at all. (**ocamorph** has built-in morphological guessing functionality, but this feature currently vastly overgenerates analyses, so we do not rely on it.)

About 3% of the tokens was OOV for our morphological analyzer. (The reader should keep in mind that the corpus consists of documents with a below-4% OOV ratio with respect to **hunspell**.) The remaining tokens fall in 195k lemmas.

#### 4.8.2 How to present the data?

Summary frequency dictionaries can be published without complications, but the publication of harvested webcorpora brings up many questions related to copyright. We address such questions below.

Clearly, the availability of large-range gigaword corpora is in the best interest of all workers in language technology, and equally clearly, only open (freely downloadable) materials allow for replicability of experiments. While it is possible to exploit search engine queries for various NLP tasks (Lapata & Keller 2004), for applications which use corpora as unsupervised training material downloadable base data is essential (Kilgariff 2007).

Our research group decided against “cover your behind” approaches such as publishing only URLs. First, URLs age very rapidly: in any given year more than 10% become stale (Cho & Garcia-Molina 2000), which makes any experiment conducted on such a basis effectively irreproducible. Second, by presenting a quality-filtered and character-set-normalized corpus the collectors actually perform a service to those who are less interested in such mundane

---

<sup>6</sup>The heuristic is unable to resolve the difference between the lexicalized *irtás irtás*/NOUN ‘glade’ and the non-lexicalized *irtás irt*/VERB [GERUND]/NOUN ‘the process of destroying something’, or more precisely, it always resolves it as the former.

issues. If everybody has to start their work from the ground up, many projects will exhaust their funding resources and allotted time before anything interesting could be done with the data. In contrast, the Free and Open Source Software (FOSS) model actively encourages researchers to reuse data.

It is important to emphasize that we do not advocate piracy: to the contrary, it is our intended policy to comply with removal requests from copyright holders, analogous to Google cache removal requests. Finally, even with copyrighted material, there are easy methods for preserving interesting linguistic data (say unigram and bigram models) without violating the interests of businesses involved in selling the running texts.

### 4.8.3 Later work

At the time of writing, our research team is in the process of creating an updated version of the Szószablya Webcorpus and frequency dictionary. As we document it in (Halácsy et al. 2008) and (Zséder et al. 2012), the data processing pipeline has been completely rewritten for this task, mainly to increase processing speed, but also with improved data quality in mind. (In particular, the document-level duplum selection algorithm was refined by the present author.) We have already used our pipeline to create and publish webcorpora and (morphologically non-disambiguated) frequency dictionaries for the following fifteen languages: Catalan, Czech, Croatian, Danish, Dutch, Finnish, Lithuanian, Norwegian, Polish, Portuguese, Romanian, Serbian, Slovak, Spanish, Swedish, with more planned. The data are made available at <http://hlt.sztaki.hu/resources/webcorpora.html>, and the source code of the text processing pipeline at <https://github.com/zseder/webcorpus>.

## 4.9 Applications

### 4.9.1 The Szószablya web-based frequency dictionary

The automatically annotated frequency dictionary has a web-based interface at the URL <http://szotar.mokk.bme.hu/szoszablya>. The interface was implemented by András Sza-

lai (unpublished). It is a tool frequently used by Hungarian linguists (Magyar & Szentgyörgyi 2011), (Rácz & Szeredi 2009), (Szeredi 2009) and psycholinguists (Racsmány et al. 2012), (Lukács et al. 2007).

To construct the application, the frequency dictionary was directly turned into an SQL database, after being augmented with two more fields: syllable count and CV skeleton. (The CV (consonant-vowel) skeleton of a word is the pattern of consonants and vowels. Examples: the CV skeleton of *hatvány* ‘power’ is *cvccvc*, the CV skeleton of *számítógépesítésének* ‘of its computerization’ is *cvvcvcvcvcvcvcvcvcvc*.) The extra fields are automatically calculated from the original ones by heuristics that work reliably for words obeying Hungarian spelling rules. The query language for the interface is just a thin wrapper around SQL query access to the tables of this database, and queries are directly translated to SQL before execution, making the query language quite versatile. The result of the query can be ordered by any of the fields. Some examples:

- lemma="nyúl" pos!="VERB" - non-verb occurrences of the lemma *nyúl* ‘rabbit’. (C.f. *nyúl* ‘to reach for’).
- word~".\*ku[^s].\*" pos=NOUN - nouns containing the character bigram *ku*, not continued by the character *s*.
- syllable\_count>=3 syllable\_count<=5 pos=VERB - verbs with syllable count between 3 and 5.
- cv\_skeleton~"cc.\*" pos!=NUM - words starting with two consonants, which are not numbers.
- analysis="NOUN<PLUR>" - plural nouns.
- lemmafreq>1000000 freq<1000 - rare words which have a frequent lemma.

### 4.9.2 Gating and prefix entropy

In this subsection we describe an application of the Szószablya frequency dictionary in the field of psycholinguistics. This is part of a joint work with Csaba Pléh, Judit Fazekas, Kornél Németh, and Klára Várhelyi (Fazekas et al. 2012), (Pléh et al. 2011). The experimental setup is based on the so-called gating paradigm (Grosjean 1980), where the participant has to recognize words from hearing fragments of them, and she hears more of the word in each successive trial.

The frequency dictionary is used to calculate entropy values for word prefixes, giving a continuous generalization and refinement of the notion that a given word prefix has a unique continuation among word stems. We show that positions with large changes in the entropy value correspond to an increased probability that the given position is the place of successful recognition.

### Gating

Gating is a traditional method to look for the temporal structure of word recognition. Since it was introduced by Grosjean a generation ago, it was used to show effects of frequency, word length, stress pattern, competing words, lexical uniqueness, morphological structure and sentential contexts (for a review see (Grosjean 1980)).

During a gating experiment, the participant has to recognize words from hearing starting segments of them, and she hears more of the word in each successive trial. In each turn, she has to provide a guess for the full word, and a confidence level in her guess.

Based on a written corpus or lexicon, we can talk about the *uniqueness point* of a given word in the corpus. This is the shortest prefix of the word that has a unique continuation in the corpus. (Namely, the given word.) There are two details to be dealt with: First, the definition is not really useful if the corpus contains inflected words, so we only work with simple words and their frequencies. Second, we do not want orthography to interfere, so we

work with phonematized data. For our purposes, we defined early uniqueness point to be earlier than 4th phoneme.

## Material

60 words were chosen for the experiments. All items were disyllabic simple nouns. 30 of them were frequent, 30 rare, and within each group 15 were used with an early uniqueness point like *japán*, and with a late uniqueness point like *cinke*. Selection was based on a manual lookup of the lemmatized Szószablya corpus, uniqueness point was calculated on a prefix tree built on the set of non-inflected words of the corpus.

In the first gating study gates with 90, 120, 210, 300, 390 ms were used. Subjects were presented with the gates, and they had to write down their responses.

A group of 51 healthy students with ages ranging from 18 to 25 years (31 female (20,41 year; SD=0,98; 20 male (21,11 year; SD=1,4)) of the Budapest University of Technology and Economics participated in this experiment as volunteers. Every subject had normal hearing. They were tested individually and they all gave informed consent before the experiment and received partial credit for participating. None of the subjects had any prior experience with the experimental task.

## Procedure

A within-subject design was used in the experiment. Every participant heard all of the word fragments with consecutively longer gates but the order of the words was randomized. After every sound segment participants had to find out what the word was and then type the total word on the computer keyboard. After typing, at every guessing they needed to assign the certainty of the answer (1 - absolutely unsure, 2 - rather unsure, 3 - rather sure, 4 - absolutely sure). If the answer was correct –irrespective of the gate time and the confidence judgment, the program presented the first fragment of the next word.



## Measuring entropy

Our earlier experiments demonstrated that the earliness of the uniqueness point is a good predictor for the point of recognition. With the entropy measures defined below we wanted to give more robust generalizations of the intuitive notion of uniqueness point.

To achieve this, we worked with the Szószablya lemmatized frequency dictionary of Hungarian. For the gating experiments, we worked with the non-inflected words of the corpus. (We repeated the following evaluations with the corpus restricted to two-syllable words, but this practically did not affect our findings.)

Each of the audio segments the subjects had to work with were manually transcribed. These transcripts are called ‘prefixes’ in the following. As an example here are the transcribed prefixes for the word *ablak*:

90ms:a 120ms:a 210ms:ab 300ms:abla 390ms:abla

Several metrics were defined over prefixes. They all measure how a word prefix constrains the set of possible continuations. The first two were only used as a baseline, and as our figures will show, their power to predict the recognition points are weaker than the entropy-based measures.

**prefixtypeoccurrenceslog** - This metric is simply the number of word forms in the corpus starting with the given prefix. This is a very skewed variable, so we take its base 2 logarithm as a normalization.

**prefixfreqlog** - This is the number of tokens in our corpus starting with the given prefix. One can consider this the weighted version of the former, weighted by word frequencies. We work with the logarithm of this quantity.

**entropy** - Entropy of the corpus, conditioned on the given prefix. Informally, this means the expected number of questions we need in a “20 Questions” game, if we must guess a randomly chosen word with the given prefix. (Randomly chosen according to the frequency distribution of the corpus.). More formally, the entropy of the observed corpus  $W$  conditioned on the prefix  $x$  is,

$$H(W|x) = \sum_{w \in W} p(w|x) \log_2 p(w|x),$$

where  $p(w|x)$  is the probability of observing the word  $w$  in the corpus, conditioned on the fact that  $w$  starts with the prefix  $x$ . (By convention,  $0 \log_2 0 = 0$ .) Typically, although not necessarily the entropy monotonically decreases when calculated for longer and longer prefixes of a given word.

**entropychange** - The fourth metric is the decrease in entropy when compared to the previous gate. This is defined as  $H(W|y) - H(W|x)$ , where  $x$  is the prefix at the current gate, and  $y$  is the prefix at the previous gate. The value is not defined for the first gate. Note that except for the first two gates, the entropy change is measured over a fix 90ms time interval.

## Results

For each event of successful recognition, the prefix at the recognition point was paired with the prefix at the immediately preceding gate (we call this one-before-recognition point). Figure 4.2 shows the means of the various prefix metrics, for both the recognition points (green) and one-before-recognition points (blue). The data for each gate were aggregated (the x axis of the graphs).

On Figure 4.3 we can see the entropy for gates after (green) versus before (blue) recognition. The four graphs control for word frequency and uniqueness point. Left-right corresponds to frequent-rare, top-bottom corresponds to early-late uniqueness.

Applying the Mann-Whitney-Wilcoxon test, we verified that the means for the recognition points differ from the means for one-before-recognition points at a high significance level.

This is true for both entropy and entropychange. We note that this result is the least obvious for the entropychange measure, as it is a highly non-monotonous function of prefix-length. Intuitively, this means that the recognition point follows a sudden drop of the entropy value, which is the hypothesis we started from. On the other hand, the confidence of the

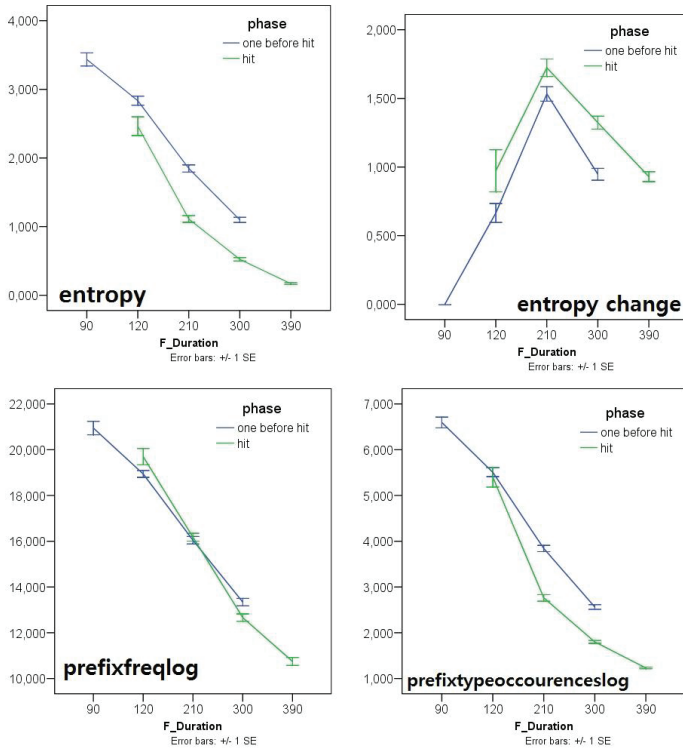


Figure 4.2: The various uniqueness metrics for recognition (green) and one-before-recognition (blue) points, averaged between items and subjects.

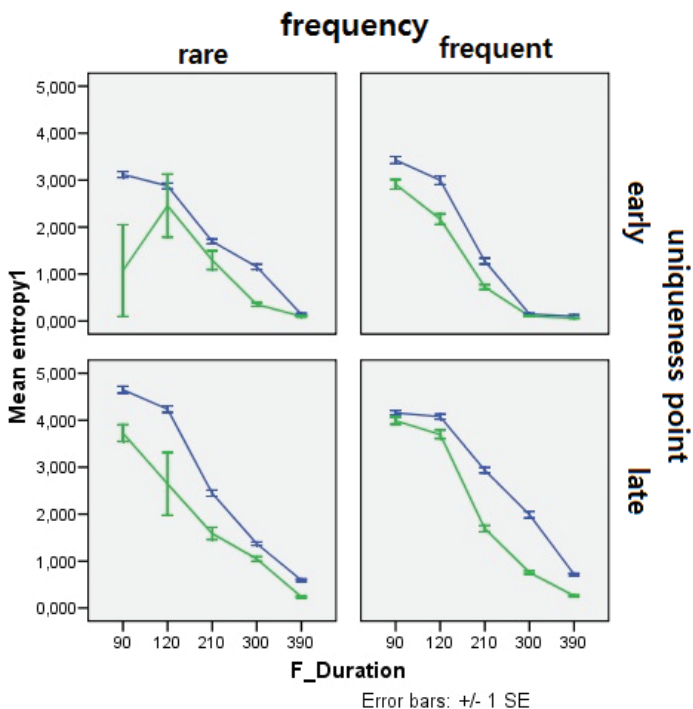


Figure 4.3: The entropy for gates after (green) versus before (blue) recognition. The four graphs control for word frequency and uniqueness point. Left-right corresponds to frequent-rare, top-bottom corresponds to early-late uniqueness.

subject in her prediction (standardized on a per-subject level) has no statistically significant correlation with the entropy change of the gate.

To falsify the hypothesis that entropy has effect only through frequency and earliness of uniqueness point, we controlled for these two binary variables. Figure 4.3 shows that entropy has significant correlation with recognition rate, even when we control for frequency and uniqueness point. We interpret this as showing that entropy is a refinement of the naive notion of uniqueness. This result is in line with the model of Moscoso, Kostic, and Baayen (Moscoso del Prado Martín et al. 2004). Namely, what is treated traditionally as a uniqueness point in gating research is actually a drop in entropy. Entropy based measures proved to be better here than mere frequency.

### **Entropy values of the word items**

As an item based post hoc test of the relevance of frequency and uniqueness a two way analysis of variance was performed over the entropy values, with frequency and uniqueness point as the two factors. Both factors were significant. For frequency  $F(1.56) = 19.213$ ,  $p < 0.001$ . Rare items had a much lower entropy at the fourth letter than frequent ones (0.844 vs 1.782). At the same time, the selection variable of late versus early decision point also had a significant effect ( $F(1.56) = 27.478$ ,  $p < 0.001$ ). The early uniqueness point items had a much lower entropy (0.752) at the fourth letter than late decision point items (1.874). This can be taken as a post hoc support of our classification. Comparison of the eta squares as an estimate of variance explained shows that decision point had a stronger effect (0.329) than frequency (0.255). While there was a significant interaction, its impact is very weak in variance explained (0.09). These estimates have to be carefully interpreted. They certainly reinforce our selection of items, but one has to bear in mind that the items were hand-picked.

## Chapter 5

### Named Entity Recognition

In the analysis of natural language text a key step is *named entity recognition*, that is, finding all complex noun phrases that denote persons, organizations, locations, and other entities designated by a name. In this chapter we introduce the **hunner** open source language-independent named entity recognition system, and present results for Hungarian.

In Section 5.1 we introduce the task of Named Entity Recognition. In Section 5.2 we describe the training corpus and other language resources used by our system. In Section 5.3 we detail the architecture and implementation of our system, and present the methodology and the results of our evaluations. In Section 5.6 we give an example of a toy data mining application built on our system.

The creation of the **hunner** system was joint work with Eszter Simon and originally appeared in (Varga & Simon 2006) and (Varga & Simon 2007). The author of the thesis is responsible for the design of the machine learning architecture, the software implementation, and the evaluation framework. Feature engineering and data collection was shared between the two original authors. The software was later reimplemented in a joint work with Gábor Recski (Recski & Varga 2009) (in Hungarian). The result of this reimplementation is the so-called **huntag** tool that is capable of named entity recognition and various chunking tasks, depending on the resources it is provided with. (We will still refer to it as **hunner** when it works with a named entity model.) The accuracy scores presented in this chapter refer to the **huntag** implementation, which did not significantly change the scores. The author was an equal collaborator in the work presented in Section 5.5, originally appearing in (Farkas

et al. 2007). The author had no significant role in the creation of the data mining example presented in Section 5.6.

## 5.1 Background

In the machine analysis of natural language documents we often seek to answer questions in terms similar to those posed by humans: *who* is this document about, *where* is the action taking place, *how much* money is involved, and so on. By *named entity recognition* (NER) we mean an algorithm that takes natural language text (typically, in document-sized chunks rather than word by word or sentence by sentence) as input, and identifies all persons, locations, organizations and similar entities that are designated by a name. The name can be a single word such as *Budapest* or a complex phrase such as *Budapesti Műszaki és Gazdaságtudományi Egyetem Média Oktató és Kutató Központ*. Even when the eventual goal is more remote (e.g. machine translation, information extraction, or information retrieval), NER is a useful intermediate stage of processing.

The trivial algorithm that identifies capitalized phrases as named entities works well for English, Hungarian, and many other languages that capitalize proper names, but of course it fails for languages like German where capitalization conventions are different, for languages like Arabic or Chinese that do not have separate upper- and lowercase, and for text that lacks casing (e.g. the output of speech recognition). Also, because the trivial algorithm is prone to false positives sentence-initially and to false negatives in text written in anything but the most carefully edited prose, it is important to develop methods that are less error-prone.

The NER task, as posed by the MUC-6, MUC-7 (Chinchor 1998) and CoNLL-2003 (Tjong Kim Sang & De Meulder 2003) competitions, is to identify disjoint chunks of the input token sequence as named entities and to annotate these chunks with a small set of named entity categories such as PERSON, ORGANIZATION, LOCATION, and MISC (all other named entities). The evaluation of an automatic NER algorithm is through comparing its output to a manual

annotation. Typically, the algorithm itself learns its parameters from a manually annotated corpus through supervised learning.

For major languages, a large amount of papers were published on NER algorithms. Almost every currently known supervised machine learning technique was used. Some examples: BBN Identifinder (Miller et al. 1998), and Zhou and Su (Zhou & Su 2002) apply Hidden Markov Modeling. Borthwick (Borthwick et al. 1998) (Borthwick 1999), and Chieu and Ng (Chieu & Ng 2003) apply maximum entropy modeling. Sekine et al. (Sekine et al. 1998) use decision trees. There are not too many language-dependent components of these and other similar systems. Still, for Hungarian, we are only aware of one quantitative study of a NER system, which is based on machine learning methods: Szarvas et al. (Szarvas, Farkas & Kocsor 2006) published results on their NER system based on C4.5 decision trees with Boosting. Their system achieved state-of-the-art accuracy for English, and for Hungarian it reached a CoNLL F-score of 94.77% on the Szeged NER Corpus (Szarvas, Farkas, Felföldi, Kocsor & Csirik 2006). (See Subsection 5.4.1 for the definition of CoNLL F-score.)

## 5.2 Resources used

### 5.2.1 The Szeged NER Corpus

For the training and evaluation of our system, we used the Szeged NER Corpus (Szarvas, Farkas, Felföldi, Kocsor & Csirik 2006). At the time of the experiments, this was the only named entity annotated Hungarian corpus large enough for supervised learning. More recently the Szeged research group created another corpus, the co-called Criminal NE corpus consisting of articles from the magazine *HVG* that are related to the topic of criminal offenses. ([http://www.inf.u-szeged.hu/rgai/nlp?lang=en&page=corpus\\_ne](http://www.inf.u-szeged.hu/rgai/nlp?lang=en&page=corpus_ne)) We have not experimented with this corpus yet, and we are not aware of published measurements on this corpus by others.

The Szeged NER Corpus is a more than 220 thousand token subset of the Szeged Corpus (Csendes et al. 2004), manually annotated for named entities. A distinct characteristic of



class	token #	phrase #
non-NE	200067	
PER	1921	982
ORG	20433	10533
LOC	1501	1294
MISC	2041	1662
all	225963	

Table 5.1: Number of tokens and phrases in the Szeged NER corpus

the Szeged NER text is its thematic homogeneity: it only contains various subgenres of business news. This means that organization names are very highly represented. This category dominates the others in frequency, see Table 5.1.

The annotation of the corpus follows the tagset and annotation conventions of CoNLL (Tjong Kim Sang & De Meulder 2003). This means that we used the following tagset: person names (PER), organization names (ORG), location names (LOC) and miscellaneous other named entities (MISC). In the Szeged NER Corpus, the MISC category mostly contains brand names and financial acronyms.

An example sentence from the corpus: „[ Sam DiPiazza ]<sub>PER</sub> , a [ PWC ]<sub>ORG</sub> vezérigazgatója szerint az [ EU ]<sub>ORG</sub> által már kötelezővé tett úgynevezett [ GAS ]<sub>MISC</sub> az eddigénél nagyobb tekintést ad az adott cég pénzügyeibe.” (According to [ Sam DiPiazza ]<sub>PER</sub> , chief executive of [ PWC ]<sub>ORG</sub> , the so-called [ GAS ]<sub>MISC</sub> , already enforced by the [ EU ]<sub>ORG</sub> , gives more insights into the finances of any given firm.)

### 5.2.2 Gazetteers

Though ‘gazetteer’ originally means geographical directory, in the context of the NER task the phrase is simply used as a list of names. We assembled various gazetteers to be incorporated into our system.

- Hungarian and common non-Hungarian last names (105418)
- Hungarian and common non-Hungarian first names (7080)
- Hungarian nicknames (2034)
- names of Hungarian cities (23840)
- country names in Hungarian (325)
- Hungarian street names (15826)
- Hungarian organization names (12595)
- international organization names (2351)
- suffixes for company names (34)
- suffixes for street names (65)
- financial acronyms (13)

In the first seven cases, our sources were an aggregated version of a Hungarian phone book, and web databases. The lists of Hungarian organization names and street names were cleaned of suffixes using automatic methods. The common suffixes (e.g. Inc., Ltd. for organizations, Street, Sq. for places, or in Hungarian ‘Kft.’, ‘Rt.’, ‘utca’, ‘tér’, respectively) were extracted, and moved into separate lists. The international organization list was kindly provided to us by György Szarvas and Richárd Farkas.

There was only one case when analyzing the development corpus led to the inclusion of a new gazetteer: the list of financial acronyms. The development corpus contained several stock market index names (e.g. DAX, Libor, Nasdaq), which were sometimes marked as **ORG** instead of **MISC** by the algorithm. To solve this problem, we extracted such stock market terms from a web-based financial knowledge base. We note that using this lexicon did not improve the performance on the test corpus, and even decreased it slightly. The reason for

this is that most of these terms occurred in the development part of the corpus, i.e. the split between the test and the development corpus was not a random one (see Sub-subsection 5.4.1 for more details).

The gazetteers incorporated into our final system (except for the list of financial terms) were finalized before the inspection of the train and development corpora. During the tuning of the system to the development corpus, we have found serious cases of over- and undergeneration in the gazetteers. Since correcting these errors did not improve accuracy significantly, we reverted to the original, uncorrected, automatically collected versions of the gazetteers, especially as this results in a cleaner methodology (less manual labor).

Similarly to the source code of the system, we have published the gazetteers under a free document license.

### **5.3 Algorithm - hunner**

As we have briefly mentioned in Subsection 2.3.2, we used a version of Maximum Entropy Markov Modeling as our supervised machine learning approach. Our system roughly follows the architecture described by (Borthwick 1999) and (Chieu & Ng 2003), incorporating some ideas introduced by (Klein et al. 2003).

#### **5.3.1 Feature extraction**

When building a supervised machine learning system, a major step is feature extraction, that is, collecting information from the raw data that can be relevant for the classification task. In the case of the NER task an obvious approach to feature extraction is to collect such information from the neighborhood of the inspected token. Some possible examples of features are capitalization, part-of-speech, or occurrence in some lexicon. The task of the supervised machine learning algorithm is then to find in this large amount of information regularities that are relevant to the classification task.

We exploit the fact that as far as CPU time and memory consumption are concerned, the maximum entropy method is capable of dealing with millions of features. Most of our features deal with very easily computable syntactic properties of tokens. On the other hand, we exploited the fact that we have the `hundisambig` morphological disambiguator at our disposal.

The feature set was composed manually. Below is the complete list of the features used by our system:

1. Is some neighborhood of the token contained in a gazetteer? If yes, is the token at beginning, ending or middle position of the phrase? (To deal with morphology, when determining the matching of multi-word phrases, we treated the last word of the phrase differently: matching on a suitably chosen prefix was enough. This corresponds to the way Hungarian multi-word phrases are inflected.)
2. Sentence start, sentence end position.
3. Boolean valued syntactic properties of the word form: upper case, all upper case, contains capitalized letter after non-capitalized (e.g. *iPad*), is a number, contains a number, contains a dash, contains a period.
4. String valued syntactic properties of the word form: The capitalization/hyphenation pattern of the word: For example, the capitalization pattern of *iPad-del* is `xxxx-xxx`. The shortened cap/hyp pattern of the word, consolidating identical runs of symbols. For the *iPad-del* example this is `xx-x`.
5. String-valued surface properties of the word form: the word form itself, the five-letter prefix, and most notably all consecutive three-letter character sequences (trigrams) of the word form. Note that we use gazetteers crafted beforehand, but in practice, these features have an effect similar to gazetteers directly extracted from the train corpus. A similar application of character n-grams for named entity recognition was first proposed by Klein et al. (Klein et al. 2003).

6. Information provided by the **hundisambig** morphological disambiguator: part-of-speech (NOUN, ART, NUM, ADJ, VERB, etc.). The lemma of the token. Is the word form recognized by **hundisambig**? Is the identified lemma differently capitalized than the token itself? What morphosyntactic features are provided by **hundisambig**?
7. Inflectional tag sequences: the concatenation of inflectional tags assigned by **hundisambig** to the tokens in a radius of 2 around the inspected token.

Example: The *Gyula* token, in a sentence starting position gets the following features:

1. Built-in Boolean features: **sentencestart caps**.
2. Character n-gram features: **tri.Gyu tri.yul tri.ula prefix.Gyula**.
3. Gazetteer features: **firstname.lone city.lone familyname.lone corp.start**. (.lone here means that the gazetteer contains the token itself, as opposed to the case of **corp.start**, which means that the **corp** gazetteer contains a phrase starting with this token.)
4. Morphological features: **postag.noun lemma.Gyula**.

The system collects these data for each individual token of a sentence. To incorporate context, we simply add the features of neighboring tokens, recording the relative positions of the tokens. For example, if a token gets the feature **caps.pre2**, it means that the token two positions before is capitalized. A parameter of our system is the size of the context window for a given feature. For simplicity's sake, we did not optimize this parameter for each feature separately. According to our experiments, in the case of character trigrams and prefixes, using context radius 2 (that is, a 5-token interval) leads to optimal results. In the case of the rest of the features, context radius 5 (11 tokens) was used.

### 5.3.2 Tag sets

The NER task in its original form deals with the classification of unknown contiguous token sequences, and it is not immediately obvious how to phrase this as a token classification task. Roughly following (Chieu & Ng 2003), we chose the following solution: every token must be classified into one of 17 different classes:  $\{ 0, \text{LOC.single}, \text{LOC.start}, \text{LOC.middle}, \text{LOC.end}, \text{ORG.single}, \dots, \text{MISC.end} \}$ . There are two major advantages of this approach: First, the machine learner can more easily recognize correlations that are specific to the start or end of the NE. Second, the tag set has implicit built-in consistency requirements: e.g. `*.start` can not follow `*.middle`. These consistency requirements can be fully learned by the tag language model.

### 5.3.3 Sequence labeling

The techniques described above allowed us to phrase the NER problem as a sequence labeling problem. As we already mentioned in Subsection 2.3.2, we solve this problem using a simplified version of Maximum Entropy Markov Modeling. In this subsection we describe this approach.

Let  $p(i, u)$  denote the probability that the word in position  $i$  receives the label  $u$ . We assume that the value of  $p(i, u)$  depends solely on the features of the words in the context  $w_{i-k} \dots w_{i+k}$ . Hence  $p(i, u)$  can be estimated by  $\hat{p}(i, u)$  supplied by a maximum entropy model trained on these features.

Let  $t(i, u, v)$  stand for the conditional probability that the word in position  $i$  receives label  $u$  providing that the word in position  $i - 1$  received the label  $v$ . We assume that this probability is independent of  $i$  and estimate it by  $\hat{t}(u, v)$ , the conditional relative frequency directly observed in the training corpus.

During labeling, the system has to find the most likely label sequence for a given sentence. If  $\hat{p}(i, u)$  only depended on  $w_i$  (no context, just the current word), then the likelihood of a

label sequence could be written as a product thanks to conditional independence, and would be proportional to

$$\prod_i \frac{\hat{p}(i, u_i) \hat{t}(i, u_i, u_{i-1})}{P(u_i)}.$$

The argmax of this formula (that is, the best labeling) can be easily found by the Viterbi algorithm. This model is in fact a variant of the ‘observations in states instead of transitions’ version of Maximum Entropy Markov Models, as suggested by (McCallum et al. 2000). Our model can be described as a theoretically unfounded simple modification of the model: we do let  $\hat{p}(i, u)$  depend on a nontrivial  $w_{i-k} \dots w_{i+k}$  ( $k > 0$ ) context, and use the above formula as an approximation of the true likelihood.

The optimum radius  $k$  of the context window was found to be 3 for these experiments. We experimented with tuning the radius parameter individually for each feature group, but the F-score gains did not justify the increased number of free model parameters.

We note that illegal transitions (e.g. `ORG.start` after `LOC.start`) are never observed in the training corpus, so transition probabilities for them are learned to be equal to zero. Unlike in the case of morphological disambiguation, the language of tagsets is quite information-poor in the case of NER, so improving the bigram tag model to a trigram model did not improve the performance of the system.

A tunable parameter of the model is language model weight (sometimes called fudge factor in speech recognition). This is a constant positive exponent applied to the transition probabilities before combining them with the emission probabilities. Setting the language model weight below 1 means that we give lower credence to the evidence obtained by the transition model than to the evidence obtained by the emission model. Experimenting with the language model weight yielded optimal values close to 1, so we used the ‘un-fudged’ value of 1 to reduce the number of manually tuned parameters of our model.

### 5.3.4 Implementation

We have chosen Zhang Le’s (Le 2011) maximum entropy learner library in our implementation. This implementation uses the L-BFGS algorithm (Zhu et al. 1997) for model parameter optimization. On our data sets, the L-BFGS iterative learning algorithm starts to converge after approximately 100 iterations. The accuracy of the model wildly fluctuates before this iteration number is reached. Our published numbers are based on 300 iterations. On the other hand, because of the relatively high (approx. 1 hour) running time of 300 iterations, some of our elementary feature engineering decisions were based on measurements with lower (30 or 100) iterations. This may have led to suboptimal decisions. We used a Gaussian prior of 1.0 for the model parameters. (Generally, the exact choice of prior only weakly influences accuracy. The value was optimized on the development corpus, chosen from exponents of 10.) Measurements on the development corpus show that overfitting is not a problem when we stay within the Szeged NER domain.

We note that using prefixes, character trigrams and wide context windows led to a very high total number of features. On the 200,000 token corpus, 1.9 million different kinds of features occur, for a total of 29.5 million feature instances. The maximum entropy approach is capable of dealing with such a high number of features without the feature selection phase needed by some other machine learning methods.

## 5.4 Evaluation

### 5.4.1 Methodology

To measure the accuracy of a machine learning algorithm, its output has to be compared to a gold standard test dataset. One standard method to quantify the similarity between two named entity labelings is the CoNLL F-score. According to this, a gold standard named entity is correctly labeled if the automatic labeling gives the same start- and end-position,



NE-type	devel	test	Szarvas et al devel	Szarvas et al test
LOC	92.06	96.36		95.07
MISC	93.58	85.12		85.96
ORG	97.62	96.20		95.84
PER	97.44	94.94		94.67
<b>Global</b>	<b>96.35</b>	<b>95.06</b>	<b>96.20</b>	<b>94.77</b>

Table 5.2: Results

and the same named entity class. Based on this, precision and recall values can be calculated for the corpus, and the F-score is, as usual, the harmonic mean of these two values.

We started the early development of our system with an ad hoc train-test split of the Szeged NER Corpus. But it quickly became apparent that if we intend our results to be comparable to the only existing quantitative study on Hungarian named entity recognition, then we will have to switch to the train-development-test split used by (Szarvas, Farkas & Kocsor 2006). Szarvas et al. were kind to provide this split, and from this point, we followed standard methodology: We optimized the parameters of the system guided by the F-score on the development corpus, and only measured the F-score on the test corpus once, when this optimization was finished.

#### 5.4.2 Results

The system described above reached an F-score of 96.35 on the development corpus, and 95.06 on the test corpus. This is a minor improvement on the numbers published by (Szarvas, Farkas & Kocsor 2006) (see Table 5.2). But we have to note that the (Szarvas, Farkas & Kocsor 2006) system was optimized in parallel for English and Hungarian. Our system needs further work to give state-of-the-art results for several languages.

We measured the effect of each major subsets of the features. As we noted, the global F-score of the system was 95.06 on the test corpus. Removing just the built-in Boolean features

(`caps`, `dash`, `shortpattern` etc.) decreased this score to 80.40. Removing just the character n-gram features decreased the score to 93.79. Removing the morphological disambiguation-based features decreased the score to 93.03. The gazetteer features had significantly less effect: removing these decreased the score to 94.80. Note that these last two sets of features were exactly the ones that required external resources. Removing both of them led to a resourceless system without seriously affecting the score: the resourceless system had an F-score of 92.87. If we remove all features that makes it possible for the system to implicitly or explicitly learn gazetteers (word forms, character n-grams, lemmata) performance drops to 87.92.

## 5.5 Metonymy resolution

In this section we briefly mention our joint result (Farkas et al. 2007) on another named entity classification task: metonymy resolution (Markert & Nissim 2007). Metonymy means using one term, or one specific sense of a term, to refer to another, related term or sense. The metonymy resolution task requires classifying potential metonymies into semantic categories such as place-for-event ‘*Vietnam was a national trauma.*’, organization-for-product, organization-for-place ‘*Woman crashes her BMW into KFC.*’.

As part of the ACL SemEval-2007 Workshop, a competition took place comparing participating systems on the task of metonymy resolution of named entities. There were six independently scored subtasks of the task, three focusing on locations, three on organizations, with three granularity levels of the output tagset for both. Among the participating systems, our system achieved the best score on all of the six subtasks (Markert & Nissim 2007). On one hand, this result required some relatively elaborate feature extraction. On the other hand, the acquired features were fed directly into an unmodified maximum entropy learner, showing that this classic method is capable of achieving state-of-the-art results given a sufficiently informative set of features.

## 5.6 Resources and applications

András Solymosi (Solymosi 2007) (in Hungarian) ran the **hunner** tool on the Origo Corpus, a large corpus consisting of news items published between 2000 and 2006 by **origo.hu**, Hungary’s largest online news portal. The size of the corpus is ~73.8 million tokens in ~3.7 million sentences, ~240,000 news items. The resulting annotated news corpus is made available in a searchable form indexed by person names at <http://solymosi.hu/origo/>.

Based on this named entity-annotated corpus, (Prekopcsák 2008) created a visualization of the co-occurrence graph of person names. (See Figures 5.1 and 5.2.) A novel idea employed in this analysis is that political affiliation of Hungarian public figures is quantitatively estimated using a PageRank-like algorithm. The algorithm assigns a real number to each node of the graph (visualized by coloring of nodes on the figures). The leader of the left-wing and right-wing political block is assigned -1 and +1 respectively, and the other nodes’ scores are calculated by an iterative force-directed process that minimizes the difference between the score of a node and the average score of its neighbors. Manual inspection showed that affiliation of political figures was identified with nontrivial precision, and public figures outside the political sphere received a neutral score.

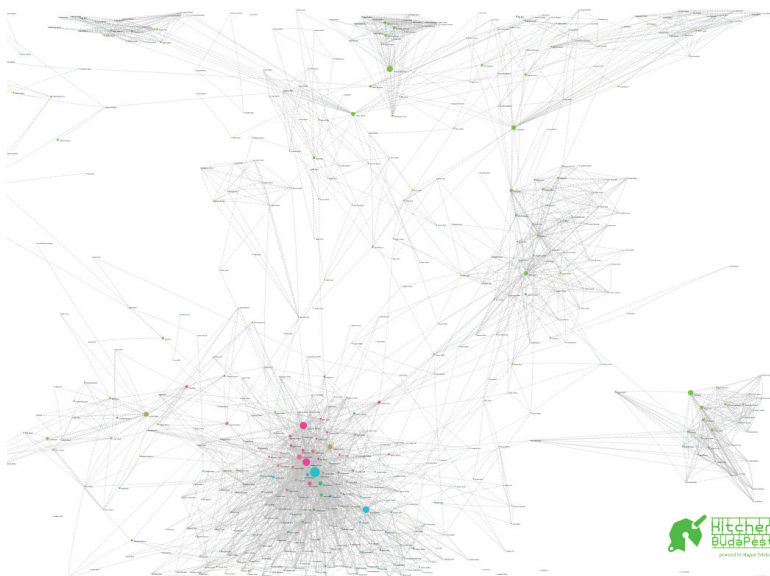


Figure 5.1: Celebgraph: a visualization of frequently mentioned Hungarian public figures based on collocation in the Origo NER Corpus.

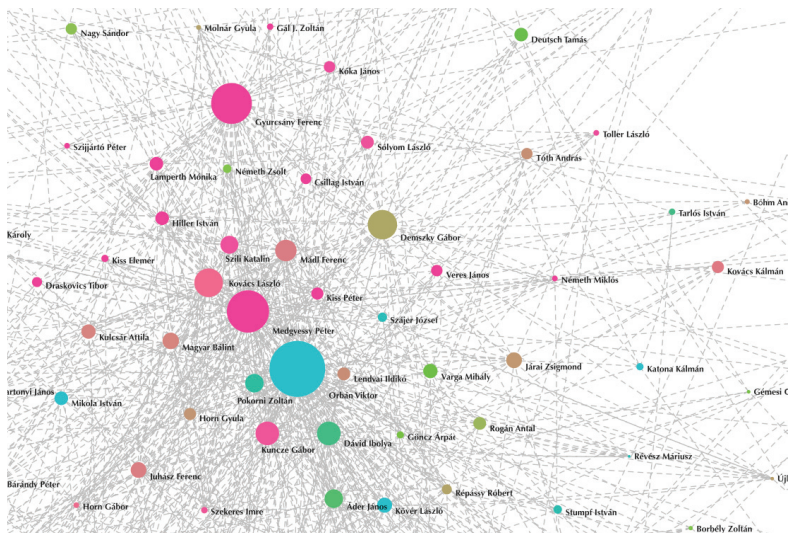


Figure 5.2: Celebgraph: Zooming in on Hungarian politicians.

## Chapter 6

### Noun phrase chunking

In the following chapter we describe a supervised Noun Phrase (NP) chunker for Hungarian. First we give a brief overview of the notion of chunks in natural language processing and describe the considerations behind the creation of the training data. Then we proceed to give a description of the chunker, and summarize the results obtained.

The results of this chapter are joint work with Gábor Recski, originally presented in (Recski & Varga 2009) and (Recski & Varga 2012) (in Hungarian).

#### 6.1 Background

(Abney 1991) describes chunks as disjoint parts of a sentence which are relevant both for language comprehension (citing (Gee & Grosjean 1983)) and sentence prosody. He defines chunks as units that consist of “*a single content word surrounded by a constellation of function words*” and claims that the internal structure of chunks can be represented by context free languages.

Abney reviews earlier definitions of chunks which called for a separate chunk for each content word in a sentence and revises it to overcome some difficulties raised by, for example, embedded adjectives. He claims a new chunk in a sentence begins after every content word except for those which are followed by another that has been selected by a preceding item. An example of the implementation of this definition is given by Abney and repeated in Figure 6.1.

This definition overcomes difficulties such as that of a noun preceded by an adjective (which occurs in Hungarian as well) yet it relies on a theoretical framework which makes use

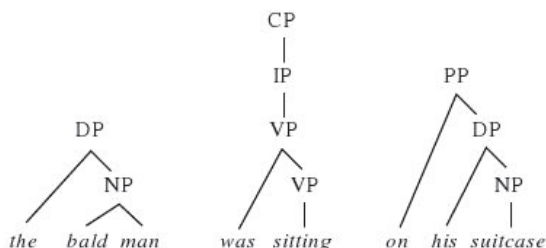


Figure 6.1: English chunk example.

of the notion of syntactic selection (we shall soon see, however, that Abney is by no means the only author suggesting a definition of NP chunks grounded in a procedural syntactic framework).

NP chunkers have been developed for various languages, most of them for English. One of the most ground-breaking efforts was that of (Ramshaw & Marcus 1995), who developed a learning algorithm which was trained on a data set derived algorithmically from a treebank and based primarily on Part-Of-Speech (POS) tags of the target data; NP chunkers have followed these conventions ever since. The article also reviews some previous approaches to the question of what to include in an NP chunk. Voutilainen (Voutilainen 1993) introduces a method for identifying NPs which do not themselves contain other NPs with the help of an extended set of POS-tags which automatically mark premodifiers of an NP as part of the chunk. Another approach is that of (Bourigault 1992), who created French NP chunks in two phases: first generating what he called “*maximal length noun phrases*” and then extracting from them so-called *terminological units*. One of the earliest results in NP chunking is that of (Church 1988) who inserted NP brackets into the POS-tagged Brown Corpus, yet he fails to provide details on how the training data was prepared, noting only that “*the training*

*material was parsed into noun phrases by laborious semi-automatic methods*". Ramshaw and Marcus later reveal that Church's parser is incapable of handling several types of complex NPs, e.g. those that contain two coordinated noun phrases. It would be a mistake, however, to compare results of the above works to each other or to those of our own since each of them refer to a slightly different and often inadequately documented task.

We define maximal NP chunks as all NPs in the syntactic tree which are not dominated by some higher level NP. It is an immediate consequence of this definition that the set of maximal NP chunks is indeed a chunking, that is, the chunks are disjoint. We will use the term *base NP* for NPs that do not contain another NP. Base NPs also constitute a chunking. In this chapter we will almost exclusively deal with maximal NP chunks, therefore unless otherwise noted, we will take the term NP chunk to mean maximal NP chunks.

For Hungarian, the first measurements on the NP chunking task are due to (Váradi 2003). He used a rule-based system, manually constructing a so-called cascaded regular grammar. The paper reports an F-score of 58.78% on a small (100 sentence, 2537 token) test corpus.

To the best of our knowledge, (Hócza 2004) was the first to publish results on supervised Hungarian NP chunking. This system learns noun phrase tree patterns described by regular expressions from the training corpus. For evaluation a corpus similar but not identical to ours was used. (See next section for more details.) Using ten-fold cross-validation the paper reports an F-score of 83%.

## 6.2 Resource used - the Szeged Treebank

To train and test a Hungarian NP chunker system, we needed some manually annotated dataset. For this, we turned to the Szeged Treebank (Csendes et al. 2005). As we briefly mentioned in Section 4.2, the Szeged Treebank adds manually built syntactic information to the Szeged Corpus, a corpus of 1.2 million tokens in 82,000 sentences. The following



enumeration of syntactic tags used by the Treebank is taken directly from (Csendes et al. 2005):<sup>1</sup>

- ADJP: adjectival phrases
- ADVP: adverbial phrases, adverbial adjectives, postpositional personal pronouns
- c: punctuation marks
- C0: conjunctions
- CP: clauses (also for marking sentences)
- INF : infinitives
- NEG: negation
- NP: noun phrases (groups with noun or predicative adjective or inflected personal pronouns as head)
- PA : adverbial participles
- PP: postpositional phrases
- PREVERB: preverbs
- V : verb
- XP: any circumstantial or parenthetic clause that is not a direct part of the sentence

The creation of the Treebank was carried out in two phases under two separate projects. In the first phase (Szeged Treebank 1.0), manual NP and CP chunking took place. In the second phase (Szeged Treebank 2.0), this was refined to a full syntactic parse, with corrections when necessary. As we will discuss, the resource we train and test our system on will be a

---

<sup>1</sup>For definitions of the terms, see e.g. (Crystal 1997).

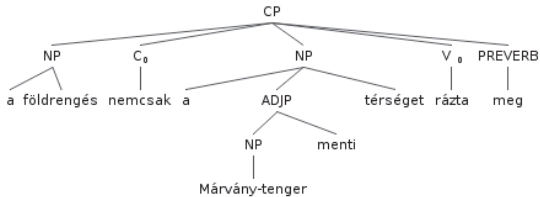


Figure 6.2: Sample sentence tree.

‘flattened’ version of the Szeged Treebank 2.0 syntactic trees, not the chunk annotation of the original Szeged Treebank 1.0. In contrast, the (Hóczka 2004) system mentioned the previous section was trained and evaluated on Szeged Treebank 1.0 data.

Since our definition of NP chunks yields phrases of various length and complexity, we defined a measure of complexity for each NP by assigning it a number that shows how many lower-level NPs it dominates. The chunking task will not involve identifying the level of an NP but the presence of this information in the training corpus may aid the machine learning task.

### 6.3 Algorithm - hunchunk

#### 6.3.1 Creating a labeling task

To solve the chunking task, we first turned it into a sequence labeling task. We marked each member of an NP with a label that indicates whether it occupies the first (B-N<sub>x</sub>), last (E-N<sub>x</sub>) or any other position (I-N<sub>x</sub>) in the chunk or whether it constitutes an NP of its own (1-N<sub>x</sub>). Here N<sub>x</sub> denotes the level of the NP. Words outside of NPs were labeled O. Therefore the sentence analyzed in the treebank as in Figure 6.2 will be labeled as in Table 6.1.

word	label
A	B-N_1
földrengés	E-N_1
nemcsak	O
a	B-N_2
Márvány-tenger	I-N_2
menti	I-N_2
térséget	E-N_2
rázta	O
meg	O

Table 6.1: The labeling corresponding to the tree on Figure 6.2.

### 6.3.2 Feature extraction

Next we proceeded to extract features from our corpus. The features of a word included its form, character trigrams and all pieces of morphological information available in the treebank. When labeling raw text, these latter features can be provided by the morphological disambiguator `hundisambig`, whose own errors, as we shall see, will only cause a slight decrease in performance.

### 6.3.3 The model

We used the same simplified MEMM sequence labeling model as the one we used for named entity recognition in the previous chapter. The optimum radius  $k$  of the context window was found to be 5, and the language model weight was set to the default value of 1.

## 6.4 Evaluation

### 6.4.1 Methodology

For the training task we used a corpus of 1 million tokens and tested the algorithm on another 100 000 tokens. We evaluated the output along the guidelines of (Tjong Kim Sang

	Precision	Recall	F-score
baseline	60.24	60.50	60.37
<b>hunchunk</b>	87.16	84.99	<b>86.06</b>
<b>hundisambig &amp; hunchunk</b>	86.19	84.20	<b>85.18</b>

Table 6.2: Evaluation of our maximal NP-chunker, with gold standard morphological disambiguation, and with one provided by **hundisambig**.

& Buchholz 2000): Precision and recall figures were calculated based on comparing the identified versus the actual set of NPs.

Note that the chunker is trained on a corpus with information about the level of NPs. This means that the chunker can provide such information for unseen data. For the purposes of the evaluation, this information was discarded.

Our baseline method was assigning a most probable label to each word based on its part-of-speech tag. Using just two labels – ‘I-NP’ for words within an NP and ‘O’ for words outside of them – we reached a baseline F-score of only 51.03%. Tweaking the system only slightly, however, – by introducing a third label, ‘B-NP’, to mark words that are at the start of an NP – increased the F-score of the baseline system to 60.37%.

## 6.4.2 Results

The results obtained are shown in Table 6.2. The last row shows the performance of the chunker when the morphological information is obtained from **hundisambig** instead of the manually annotated Szeged Treebank.

(Miháltz 2011) evaluated **hunchunk** together with two rule-based NP-chunkers: a rule-based NP-chunker system for Hungarian (Váradí & Gábor 2004), and the Hungarian syntactic parser used by the MetaMorpho machine translation system (Prószéky et al. 2004). We note that the methodology is heavily biased towards **hunchunk**, as the test corpus is

	Precision	Recall	F-score
<b>hunchunk</b>	78.67	84.99	81.71
<b>MetaMorpho</b>	54.39	61.52	57.73
<b>NooJ</b>	37.57	59.28	45.99

Table 6.3: Comparison with two Hungarian rule-based maximal NP-chunkers, taken from (Miháltz 2011).

based on a subset of the Szeged Corpus, albeit a subset disjoint from our training corpus. As Table 6.3 shows, **hunchunk** achieves higher scores than the rule-based systems.

As we already noted, (Hócza 2004) reports an F-score of 83% with a rule-learner algorithm on a corpus similar to ours, using tenfold cross-validation. Although our corpora are not completely comparable, we also performed tenfold cross-validation on our corpus, achieving an F-score of 89.30% (precision 89.75%, recall 88.86%).

## 6.5 Later work and applications

Co-author Gábor Recski extended the above results in various ways (Recski 2010). First, by training and evaluating the **hunchunk** system on base NPs. Base NP chunking is an easier task than maximal NP chunking, which is reflected in the fact that the base NP chunker achieved an F-score of 94.75%. (For comparison, the maximal NP chunker achieves an F-score of 86.06% on the maximal NP chunking task, when the test sets for the two tasks were created from the same test treebank.)

Recski also implemented the context-free grammar of Hungarian base NPs written by András Kornai (Kornai 1985, Kornai 1989). The rule-based system was not competitive with statistical systems in itself, achieving an F-score of 89.36% on the base NP chunking task. But when its output was presented to the statistical system as an extra feature, this increased

the accuracy of the statistical system, leading to a state-of-the-art F-score of 95.48% for the base NP chunking task.

(Recski et al. 2010) presented a system aligning maximal NPs for Hungarian-English. The system relies on **hunchunk** for the chunking task, and uses various heuristics to propagate GIZA++ (Och & Ney 2003) word alignments to the NP level.

## Chapter 7

### Sentence Alignment

Modern methods of machine translation and automated dictionary- and thesaurus building take as their input parallel corpora which contain the same material in two languages. *Sentence alignment*, finding which sentence in the source language corresponds to which sentence in the target language is a low-level task that is key to preparing the data for machine translation and related tasks. In this chapter we introduce and evaluate the **hunalign** sentence alignment algorithm, its **partialAlign** companion, and document the construction of several parallel corpora that were sentence aligned with these tools.

The design, implementation, and evaluation of **hunalign** and **partialAlign** are the work of the present author. The creation of the corpora was a collaborative effort in each case. In the case of the Hunglish Corpus (Varga et al. 2005), the majority of work was done by the author, from data cleaning to packaging. In the case of the JRC-Acquis Corpus (Steinberger et al. 2006), the author was responsible for the automatic sentence alignment of sentence-segmented data.<sup>1</sup>

#### 7.1 Background

A parallel text is defined as some text together with its translation to another language. (Sometimes translations to several languages are considered. We follow the majority of the current literature and focus on the bilingual case, reducing the multilingual case to working

---

<sup>1</sup>Work on **hunalign** and the original Hunglish Corpus was part of the Hunglish project, an ITEM grant supported by the the Hungarian Ministry of Informatics and Communication. Work on Hunglish Corpus 2.0 was supported by the CLARIN project at the Research Institute for Linguistics at the Hungarian Academy of Sciences.

with many language pairs.) One document and its translation is called a bitext. A collection of bitexts is called a parallel corpus. Parallel corpora are very important resources in modern natural language processing. For example, the parallel corpus is the main or sometimes even sole language resource used when building a current statistical machine translation system (Koehn 2010).

Usually, the original is called source text, and the translation is called target text. We will not focus on this distinction below, as the algorithms we will use can not and need not treat source and target texts differently.

If two documents can be considered translations of each other, normally there are correspondences between units of the two documents, at various levels of text units. The manual or automatic identification of these correspondences is called alignment. Alignment tasks include paragraph alignment, sentence alignment, phrase alignment and so on, depending on which kind of unit we are focusing on. Obviously, translations are never perfectly literal, and the differing grammars and idiom-sets of the two languages make it impossible to have perfect one-to-one correspondence between units, especially at the finer granularity levels. Even if we allow many-to-one or even many-to-many correspondences, the correspondence will be incomplete at the phrase and word levels. In practice, we can find artifacts damaging the correspondence even at the coarsest granularities. While building a large parallel corpus of novels, we observed prefaces by the translator, footnotes, and even drastic simplifications for a younger audience by the translator.

Gale and Church in their seminal paper (Gale & Church 1991) call a correspondence between parts of the two texts an *alignment* if it is monotone with respect to word (sentence, paragraph, etc.) order. More recent literature uses the term alignment even if the correspondence is not monotone. In this chapter we will focus on the task of finding corresponding parts at the sentence level. At this level, the assumption of monotonicity is a very good approximation, as translators seldom change sentence order. We will work with this assumption, so calling the task sentence alignment is justified even according to Gale and



Church’s terminology. In this chapter, the term ‘alignment’ is always used in the stricter sense. The simplifying assumption of monotonicity is very fruitful for algorithmic reasons: if we find a corresponding sentence pair in the bitext, then the pair splits the task into two smaller, completely independent subproblems: finding the alignment above the sentence pair, and finding the alignment below them. Problems amenable to this sort of divide-and-conquer can generally be solved using dynamic programming, and this is what our `hunalign` sentence aligner employs.

There are two main sources of the fact that correspondences are not always one-to-one: first, during translation sentences can coalesce, and sentences can split into two or more sentences. Second, sometimes a sentence or block of sentences is simply not translated (deletion), or is inserted into the translation (insertion). Relying on the monotonicity assumption, the alignment of a text can be represented by what we call a ladder, i.e. an array of pairs of sentence boundaries: rung  $(i, j)$  is present in the ladder if and only if the first  $i$  sentences of the source text correspond to the first  $j$  sentences of the target text. The ‘space between two consecutive rungs’ is typically one-to-one, that is  $(i, j)$  is followed by  $(i + 1, j + 1)$ . Coalescing and merging sentences can be interpreted naturally in this formalism. The case of deleted or inserted sentences is a bit more complex: obviously, a single deleted sentence corresponds to consecutive rungs  $(i, j)$  and  $(i + 1, j)$ , but we note that this correspondence is not unique if the two sides are not in valid correspondence.<sup>2</sup> So the above formalism is insufficient to properly handle the case when two blocks of sentences are not in valid correspondence with each other. Most sentence alignment algorithms do not attempt to handle this case, or only handle it implicitly. To deal with this case, an algorithm needs to output confidence scores for the holes between consecutive rungs. Let us take the example of a 5000-sentence book starting with a 100 sentence foreword on both sides, but with completely different forewords for the two languages. Let us call a ladder *proper* if it does not split blocks of non-corresponding sentences into more parts. The proper ladder in this case would

---

<sup>2</sup>A bitext consisting of a single incorrectly translated sentence can be described by either the ladder  $[(0, 0), (0, 1), (1, 1)]$  or the ladder  $[(0, 0), (1, 0), (1, 1)]$ .

be  $(0, 0), (100, 100), (101, 101), \dots, (5000, 5000)$ , with a low confidence score assigned to the first hole. (In the example above, a ladder starting with  $(0, 0), (40, 30), (100, 100)$  would be improper.) We define the sentence alignment task as finding a proper ladder with the highest possible granularity, with low confidence score assigned to holes where the two blocks are non-corresponding.

## 7.2 Algorithm - hunalign

In this section we describe our **hunalign** sentence aligner algorithm and implementation, and provide evaluations of its accuracy and run-time performance.

### 7.2.1 The algorithm

There are three main approaches to the problem of corpus alignment at the sentence level: length-based (Brown et al. 1991, Gale & Church 1991), dictionary- or translation based (Moore 2002, Chen 1993, Melamed 2000), and partial similarity-based (Simard et al. 1998).

This last method in itself may work well for Indo-European languages (probably better between English and Romanian than English and Slovenian), but for Hungarian the lack of etymological relation suggests that the number of cognates will be low. Even where the cognate relationship is clear, as in *computer/kompjűter*, *strike/sztrájk* etc., the differences in orthography make it hard to gain traction by this method. Therefore, we chose to concentrate on the dictionary and length-based methods, and designed a hybrid algorithm, **hunalign**, that amalgamates the two.

For the description of the algorithm, let us first assume that we have a bilingual lexicon at our disposal. (Later we will see how to get rid of this assumption without any serious decrease of accuracy.) In the first step of the alignment algorithm, the bilingual lexicon is used to create a crude translation of the source text by converting each word token into the dictionary translation that has the highest frequency in the target corpus, or to itself in case of lookup failure.

This pseudo target language text is then compared against the actual target text on a sentence by sentence basis. The similarity score between a source and a target sentence consists of two major components: token-based and length-based. The dominant term of the token-based score is the number of shared words in the two sentences, normalized with the larger token count of the two sentences. A separate reward term is added if the proportion of shared numerical tokens is sufficiently high in the two sentences. (This term is especially useful for the alignment of legal texts).

For the length-based component, the character counts of the original texts are incremented by one, and the score is the ratio of the two numbers. We can compare this with the Gale-Church solution of modeling target sentence length as a multiple of source target length plus Gaussian noise. (In the original Gale-Church algorithm, the variance of the noise is learned from paragraph lengths, which means that the algorithm requires paragraph segmented input to work. In its most widely used `Vanilla` implementation, the variance of the noise is hardwired.) We found that our simpler formula is faster to calculate and more robust to outliers.

The relative weight of the components was set so as to maximize precision on the Hungarian–English training corpus, but seems a sensible choice for other languages as well. Paragraph boundary markers are treated as sentences with special scoring: the similarity of two paragraph-boundaries is a high constant, the similarity of a paragraph-boundary to a real sentence is minus infinity, so as to make paragraph boundaries pair up.

The similarity score is calculated for every sentence pair around the diagonal of the alignment matrix (at least a 500-sentence neighborhood is calculated or all sentences closer than 10% of the longer text). This is justified by the observation that the beginning and the end of the texts are considered aligned and that the sentence ratio in the parallel text represents the average one-to-many assignment ratio of alignment segments, from which no significant deviations are expected. We find that 10% is high enough to produce reassuringly high recall figures even in the case of faulty parallelism such as long surplus chapters.

Once the similarity matrix is obtained for the relevant sentence pairs, the optimal alignment trail is selected by dynamic programming, going through the matrix with various penalties assigned to skipping and coalescing sentences. The score of skipping (1-to-0 and 0-to-1 correspondence) is a fixed parameter, learned on our training corpus. The score of coalescing (1-to-2 and 2-to-1 correspondence) is the length-based score of the concatenation of the two sentences plus the worse of the two token-based scores. For performance reasons, the dynamic programming algorithm does not take into account the possibility of more than two sentences matching one sentence. After the optimal alignment path is found, a post-processing step iteratively coalesces a neighboring pair of one-to-many and zero-to-one segments wherever the resulting new segment has a better character-length ratio than the starting one. With this method, any one-to-many segment can be discovered.

The hybrid algorithm presented above remains completely meaningful even in the total absence of a dictionary. In this case, the crude translation will be just the source language text, and sentence-level similarity falls back to surface identity of words.

After this first phase a simple dictionary can be bootstrapped on the initial alignment. From this alignment, the second phase of the algorithm collects one-to-one alignments with a score above a fixed threshold. Based only on all one-to-one segments, co-occurrences of every source-target token pair are calculated. These, when normalized with the maximum of the two tokens' frequency yield an association measure. Word pairs with association higher than 0.5 are used as a dictionary. The dictionary can then be used in a second round of alignment, in addition to the originally provided dictionary resource if there was one, or in a completely unsupervised fashion when such a resource was not available.<sup>3</sup>

---

<sup>3</sup>For the dictionary-building step, we also implemented several variants of the competitive linking algorithm first proposed by Melamed (Melamed 1998). These led to higher quality lexicons collected, but according to our measurements, this did not translate to improvements in sentence alignment in the third phase, so for performance reasons we used the simplest technique.

### 7.2.2 Related work - the BSA algorithm

Our algorithm is similar in spirit to Moore’s BSA (Moore 2002) in that they both combine the length-based method with some kind of translation-based similarity. In what follows we discuss how the BSA algorithm differs from ours.

BSA also has three phases. First, an initial alignment is computed based only on sentence length similarity. Next, an IBM ‘Model I’ translation model (Brown et al. 1993) is trained on a set of likely matching sentence pairs based on the first phase. Finally, similarity is calculated using this translation model, combined with sentence length similarity. The output alignment is calculated using this complex similarity score. Computation of similarity using Model I is rather slow, so only alignments close to the initially found alignment are considered, thus restricting the search space drastically.

Our simpler method using a dictionary-based crude translation model instead of a full IBM translation model has the very important advantage that it can exploit a bilingual lexicon (if one is available) and tune it according to frequencies in the target corpus, or even enhance it with extra local dictionary bootstrapped from an initial phase. BSA offers no such way to tune a preexisting language model. This limitation is a real one when the corpus, unlike the news and Hansard corpora more familiar to those working on high density languages, is composed of very short and heterogeneous pieces. In such cases, as in web corpora, movie captions, or heterogeneous legal texts, average-based models are actually not close to any specific text, so Moore’s workaround of building language models based on 10,000 sentence subcorpora has little traction.

On top of this, our translation similarity score is very fast to calculate, so the dictionary-based method can be used already in the first phase where a much bigger search space can be traversed. If the lexicon resource is good enough for the text, this first phase already gives excellent alignment results.

Maximizing alignment recall in the presence of noisy sentence segmentation is an important issue, particularly as language density generally correlates with the sophistication of

NLP tools, and thus lower density implies poorer sentence boundary detection. From this perspective, the focus of **BSA** on one-to-one alignments is less than optimal, since excluding one-to-many and many-to-many alignments may result in losing substantial amounts of aligned material if the two languages have different sentence structuring conventions.

While speed is often considered a mundane issue, **hunalign**, written in C++, is significantly faster than **BSA** (written in Perl), and the increase in speed can be leveraged in many ways during the building of a parallel corpus with tens of thousands of documents. First, rapid alignment allows for more efficient filtering of texts with low confidence alignments, which usually point to faulty parallelism such as mixed order of chapters (as we encountered in *Arabian Nights* and many other anthologies), missing appendices, extensive extra editorial headers (typical of Project Gutenberg), comments, different prefaces in the source texts etc. Once detected automatically, most cases of faulty parallelism can be repaired and the texts realigned. Second, debugging and fine-tuning lower-level text processing steps (such as the sentence segmentation and tokenization steps) may require several runs of alignment in order to monitor the impact of certain changes on the quality of alignment. This makes speed an important issue. Interestingly, runtime complexity of **BSA** seems to be very sensitive to the faults in parallelism. Adding a 300 word surplus preface to one side of *1984* but not the other slows down this program by a factor of five, while it has no detectable impact on **hunalign**. (See more about this in Subsection 7.2.4.)

Finally, **BSA**, while open source and clearly licensed for research, is not free software. In particular, parallel corpora aligned with it can not be made freely available for commercial purposes. Since we wanted to make sure that the corpora we build with our tools are available for any purpose, including commercial use, **BSA** was not a viable choice for us.

### 7.2.3 Evaluating accuracy

In this section we describe our attempts to assess the quality of our parallel corpus by evaluating the accuracy of the sentence aligner on texts for which manually produced alignment is available. We also compare our algorithm to BSA.

Evaluation shows **hunalign** has high precision: generally it aligns incorrectly at most a handful of sentences. As measured by Moore’s method of counting only on one-to-one sentence-pairs, precision and recall figures in the high nineties are common. But these figures are overly optimistic because they hide one-to-many and many-to-many errors, which actually outnumber the one-to-one errors. In 1984, for example, 285 of the 6732 English sentences or about 4.3% do not map on a unique Hungarian, and 716 or 10.6% do not map on a unique Romanian sentence – similar proportions are found in other alignments, both manual and automatic.

To take these errors into account, we used a slightly different figure of merit, defined as follows. The alignment trail of a text can be represented by a ladder, i.e. an array of pairs of sentence boundaries: rung  $(i, j)$  is present in the ladder if and only if the first  $i$  sentences on the left correspond to the first  $j$  sentences on the right. Precision and recall values are calculated by comparing the predicted and actual rungs of the ladder: we will refer to this as the *complete rung* count as opposed to the *one-to-one* count. In general, complete rung figures of merit tend to be lower than one-to-one figures of merit, since the task of getting them right is more ambitious: it is precisely around the one-to-many and many-to-one segments of the text that the alignment algorithms tend to stumble.

Table 7.1 presents precision and recall figures based on all the rungs of the entire ladder against the manual alignment of the Hungarian version of Orwell’s 1984 (Dimitrova et al. 1998).

If length-based scoring is switched off and we only run the first phase without a dictionary, the system reduces to a purely identity based method we denote by *id*. This will still often produce positive results since proper nouns and numerals will “translate” to themselves.

condition	precision	recall
<i>id</i>	34.30	34.56
<i>id+swr</i>	74.57	75.24
<i>len</i>	97.58	97.55
<i>len+id</i>	97.65	97.42
<i>len+id+swr</i>	97.93	97.80
<i>dic</i>	97.30	97.08
<i>len+dic-stem</i>	98.86	98.88
<i>len+dic</i>	99.34	99.34
<i>len+boot</i>	99.12	99.18

Table 7.1: Accuracy of the sentence-level aligner under various conditions

With no other steps taken, on *1984 id* yields 34.30% precision at 34.56% recall. By the simple expedient of stop-word removal, *swr*, the numbers improve dramatically, to 74.57% precision at 75.24% recall. This is due to the existence of short strings which happen to have very high frequency in both languages (the two predominant false cognates in the Hungarian-English case were *a* ‘the’ and *is* ‘too’).

Using the length-based heuristic *len* instead of the identity heuristic is better, yielding 97.58% precision at 97.55% recall. Combining this with the identity method does not yield significant improvement (97.65% precision at 97.55% recall). If, on top of this, we also perform stop-word removal, both precision (97.93%) and recall (97.80) improve.

Given the availability of a large Hungarian-English dictionary created by Attila Vonyó (Halácsy, Kornai, Németh, Sass, Varga, Váradi & Vonyó 2005) (in Hungarian), we also established a baseline for a version of the algorithm that makes use of this resource. Since the aligner does not deal with multiword tokens, entries such as *Nemzeti Bank* ‘National Bank’ are eliminated, reducing the dictionary from about 250k to about 120k records. In order to harmonize the dictionary entries with the lemmas of the stemmer, the dictionary is also stemmed with the same tool as the texts. Using this dictionary (denoted by *dic* in



the Table) without the length-based correction results in slightly worse performance than identity and length combined with stop-word removal.

If the translation-method with the Vonyó dictionary is combined with the length-based method (*len+dic*), we obtain the highest scores 99.34% precision at 99.34% recall on rungs (99.41% precision and 99.40% recall on one-to-one sentence-pairs). In order to test the impact of stemming we let the algorithm run on the non-stemmed text with a non-stemmed dictionary (*len+dic-stem*). This established that stemming has indeed a substantial beneficial effect, although without it we still get better results than any of the non-hybrid cases.

Given that the dictionary-free length-based alignment is comparable to the one obtained with a large dictionary, it is natural to ask how the algorithm would perform with only the bootstrapped dictionary. With no initial dictionary but using this automatically bootstrapped dictionary in the second alignment pass, the algorithm yielded results (*len+boot*), which are, for all intents and purposes, just as good as the ones obtained from combining the length-based method with our large existing bilingual dictionary (*len+dic*). This is shown in the last two lines of Table 7.1.

To summarize our results so far, the pure sentence length-based method does as well in the absence of a dictionary as the pure matching-based method does with a large dictionary. Combining the two is ideal, but this route is not available for the many medium density languages for which bilingual dictionaries are not freely available. However, a core dictionary can automatically be created based on the dictionary-free alignment, and using this bootstrapped dictionary in combination with length-based alignment in the second pass is just as good as using a human-built dictionary for this purpose. In other words, the lack of a high-quality bilingual dictionary is no impediment to aligning the parallel corpus at the sentence level.

While we believe that an evaluation based on all the rungs of the ladder gives a more realistic measure of alignment accuracy, for the sake of correct comparison with **BSA**, we present some results using the one-to-one alignments metric. Table 7.2 summarizes results on

task	hunalign		BSA	
	prec	rec	prec	rec
<i>1984-HE-S</i>	99.22	99.24	99.42	98.56
<i>1984-HE-U</i>	98.88	99.05	99.24	97.39
<i>1984-RE-U</i>	97.10	97.98	97.55	96.14
<i>CoG-HE-S</i>	97.03	98.44	96.45	97.53

Table 7.2: Comparison of **hunalign** and **BSA** algorithm on three texts. Accuracy scores are based on one-to-one alignments only.

Orwell’s *1984* for Hungarian–English (*1984-HE-S*, stemmed and *1984-HE-U*, unstemmed), Romanian–English (*1984-RE-U*, unstemmed), as well as on Steinbeck’s *Cup of Gold* for Hungarian–English (*CoG-HE-S*, 80k words, stemmed) using **hunalign** (with bootstrapped dictionary, no further tuning and omitting paragraph information) and **BSA** (with the default values).

In order to be able to compare the Hungarian and Romanian results for *1984*, we provide the Hungarian case for the unstemmed *1984*. One can see that both algorithms show a drop of precision. This makes it clear that the drop in quality from Hungarian–English to Romanian–English can not be attributed to the fact that we tuned our system on the Hungarian case. As mentioned earlier, the Romanian translation has 716 non-one-to-one segments compared to the Hungarian translation’s 285. Given both algorithm’s preference to globally diagonal and locally one-to-one alignments, this difference in one-to-one alignments is likely to render the Romanian–English alignment a harder task.

In order to sensibly compare our results with that of **BSA**, paragraph information was not exploited. Our **huntoken** sentence tokenizer is able to identify paragraph boundaries which are then used by the aligner. Experiments showed that paragraph information can

substantially improve alignment scores: measured on the Hungarian–English alignment of Steinbeck’s ‘Cup of Gold’, the number of incorrect alignments drop from 148 to 115.<sup>4</sup>

After the publication of our results, (Krynicky 2006) compared three aligners for Polish–English text for several corpora and lemmatization setting, finding that *BSA* often outperformed *hunalign* and *GMA* (Melamed 1998) in precision, but *hunalign* always outperformed the other two aligners in F-scores. (Here F-score was measured on the set of one-to-one correspondences.)

#### 7.2.4 Evaluating execution time

Very recently a paper compared speed and memory consumption of various aligners (Toral et al. 2012). Arguably such an endeavor has limited utility without corresponding comparisons in accuracy, as most such tools have tunable parameters that can lead to a broad range of accuracy-precision trade-offs. Nevertheless it is meaningful to ask performance numbers about the ‘stock’, unmodified setup of the tools on some standard dataset.<sup>5</sup> The numbers reported by (Toral et al. 2012) differed from our own measurements by a non-negligible amount. For example, on 5000 sentence documents the paper reports *hunalign* to be approximately 5 times faster than *BSA*, while our ad hoc measurements on the Hunglish Corpus (see next section) suggested an execution time ratio above 10. The sources of this discrepancy turned out to be related to the ‘incremental widening optimization’ employed by *BSA*: the algorithm first tries to find the optimal path in a very narrow band around the diagonal of the similarity matrix. If the path found touches the boundary of the band, then the path is suspect. In this case, the band is widened and the similarity values and path are recalculated. This process is repeated until a satisfying path is found. The advantage of this optimization

---

<sup>4</sup>Although paragraph identification itself contains errors, improvement may be due to the fact that paragraphs, however faulty, are consistent in terms of alignment.

<sup>5</sup>Or in the case of *hunalign*, the ‘usual’ setup as opposed to the ‘stock’ setup: By default, *hunalign* does not use the three-phase realignment procedure, only a single phase, but users of *hunalign* typically employ realignment. This approximately triples running time and memory consumption. In this subsection we will report performance numbers for the setup with realignment.

is that under good circumstances, similarity values only have to be calculated in a narrow band. The disadvantage is that under bad circumstances, the first (shorter) iterations of this process are increasing execution time without adding any benefit.

One source of the discrepancy between our measured execution times and those of (Toral et al. 2012) is that the authors of the paper made a decision that arguably led to a major methodological problem: they used already perfectly aligned text as input. The input bicorpus was built by taking bisentences from the aligned Spanish-English EUROPARL corpus (Koehn 2005), and merging them into a bitext. This meant that the alignments to be found consisted of 1-to-1 matchings only. (Not counting the small number of cases where the automatic aligner used to build the EUROPARL corpus made a mistake.) The optimal path was thus very close to the diagonal, and the **BSA** algorithm successfully terminated even before the first round of incremental widening.

To quantify the bias caused by this methodological problem, we created a comparable corpus with more realistic characteristics. We used the unaligned Spanish-English EUROPARL corpus for this. Our subcorpus has two, non-independent parts. One consists of 100 randomly chosen documents, each containing approximately 5000 sentences on the Spanish side. The second is a pairing of the above documents, to get 50 documents, each containing approximately 10000 sentences. (This pairing step was necessary because the EUROPARL corpus does not contain documents significantly larger than 5000 sentences.)

Figure 7.1 shows a scatterplot of execution times by **hunalign** and **BSA** on this corpus. The  $x$  axis shows sentence counts on the Spanish side. (The English sentence counts are similar.) The  $y$  axis shows execution times in seconds. Dots correspond to documents: red dots to **hunalign** results, blue dots to **BSA** results. The ratio of average execution times is 6.67 for the ~5000-sentence subcorpus and 5.79 for the ~10000-sentence subcorpus. The distribution of execution times for **BSA** is highly skewed, reflecting the strong dependence on the number of incremental widening rounds. This suggests reporting median numbers

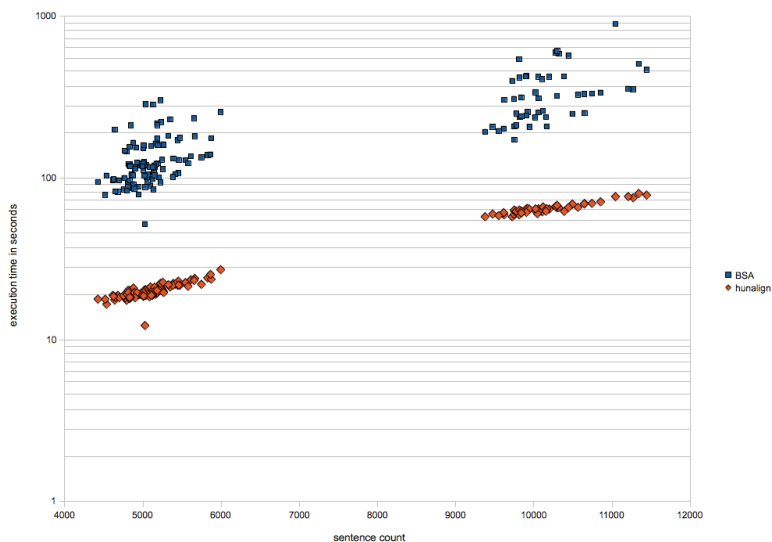


Figure 7.1: Comparing execution times of BSA and `hunalign` on ~5000 and ~10000 sentence documents from the unaligned EUROPARL corpus. Execution times shown on a logarithmic scale.

in addition to averages: the ratio of median execution times is 6.12 for the ~5000-sentence subcorpus and 5.05 for the ~10000-sentence subcorpus.

The execution time ratios reported above are still quite far from the aforementioned values measured on the Hunglish Corpus (between 5 and 15). The main cause of this remaining discrepancy can be seen on Table 7.3. The Table shows the percentage of the different rung types output by BSA, calculated on the ~5000-sentence EUROPARL and Hunglish subcorpora. The rung types in BSA terminology are match (1-1), contract (2-1), expand (1-2), delete (1-0), and insert (0-1). Although the ratio of merging and splitting sentences

Rung type	EUROPARL	Hunglish
match	91.45	75.75
contract	3.77	3.00
expand	2.10	1.57
delete	1.25	8.88
insert	1.43	10.79

Table 7.3: Percentage of rung types, as determined by BSA.

is similar for the EUROPARL and Hunglish corpora, the number of deleted and inserted sentences differs greatly. The reason for this is that unlike the very uniform EUROPARL texts, the Hunglish texts typically include parts such as title page, copyright page, table of contents, foreword, and footnotes. The content of these (and whether they are present at all) can be very different between the two sides of the bitext.

### 7.3 Algorithm - `partialAlign`

As we noted, `hunalign`'s running time and memory consumption becomes quite high above the 20000 sentences per language range. There are documents larger than that, for example J.K. Rowling's "*Harry Potter and the Goblet of Fire*" consists of approx. 21000 sentences. To overcome this limitation, we have created a companion software for `hunalign`. `partialAlign` takes a large bidocument and splits it into pieces more manageable for `hunalign`. It is designed to be especially convenient to use in tandem with `hunalign`, but we note that it works just as well when coupled with some other sentence aligner.

`partialAlign` works by finding words that occur exactly twice in the bidocument, once on the left and once on the right side. A typical example of such a word is a chapter number, other number, or a proper name. A typical literary text in our Corpus (see next section) contains one such correspondence per every 200 sentences on average. (Most non-literary

texts contain much more numberings and named entity hapaxes, so for example the ~11000-sentence EU Constitution contains 497 correspondences.)

After collecting all correspondences, the algorithm uses dynamic programming to find the longest possible chain of correspondences that does not contain crossings. (A crossing is an incompatible pair of correspondences that can not be refined into an alignment.) We call such a chain a ladder.

False positive correspondences are relatively rare. (Indirect evidence for this claim is that for a typical text in our Corpus, more than 90% of the correspondences are included in the longest ladder.) The high frequency of the correspondences means that the no-crossing restriction quite severely constrains the set of possible ladders. The above two considerations imply that a false positive correspondence is excluded from the longest ladder with high probability, thanks to the nearby true positive correspondences that override its false evidence. On Figure 7.2 we can see the scatterplot of correspondences on concatenated works of J.K. Rowling (~55000 sentences for each language). The algorithm finds 127 correspondences, and drops 6 of these.

In the final stage of processing, the algorithm throws away rungs of the ladder to approximately obey the user-provided constraint on chunk size. Again, thanks to the high frequency of correspondences, this approximation is typically good.

Unlike `hunalign`, the running time of `partialAlign` is negligible. Splitting the above ~55000-sentence corpus into ~5000-sized chunks took 2 seconds on our computer.

## 7.4 Resource built - the Hunglish Corpus

In this section we describe the creation of a 115 million token Hungarian-English parallel corpus we call the Hunglish Corpus as a case study for building parallel corpora for medium density languages. We will refer to two different versions of the corpus in the text: the original, 60 million token Version 1.0 of the corpus was documented in (Varga et al. 2005). For the creation of Hunglish Corpus Version 2.0 the data processing pipeline was fully rebuilt under

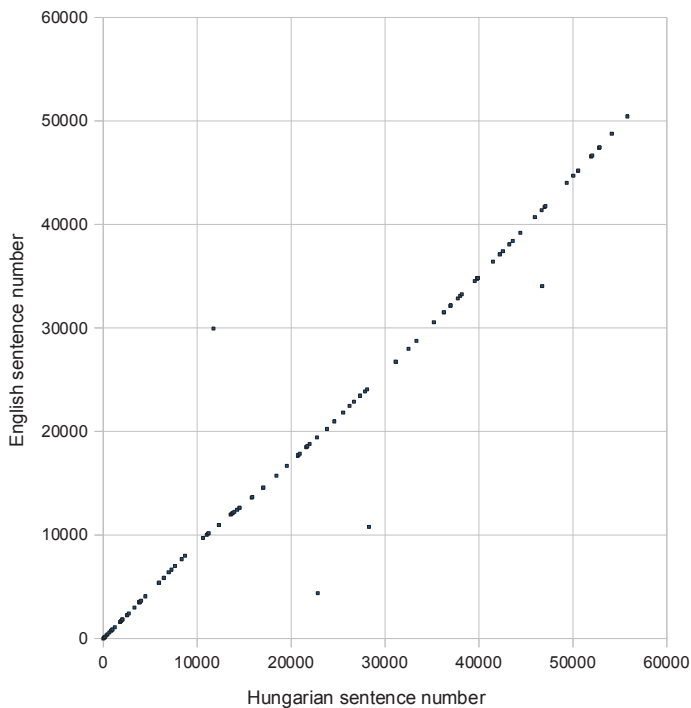


Figure 7.2: Scatterplot of correspondences for Rowling subcorpus.

the `huntools` multilingual text processing framework. (See Chapter 8 for a description of the framework.) This code consolidation did not affect major functionality, but added minor features such as automatic language checking and character encoding correction.



### 7.4.1 Collecting the data

Starting with Resnik (Resnik 1998), mining the web for parallel corpora has emerged as a major technique, and between English and another high density language, such as Chinese, the results are very encouraging (Chen & Nie 2000, Resnik & Smith 2003). Web pages are undoubtedly valuable for a diversity of styles and contents that is greater than what could be expected from any single source. However, when no highly bilingual domain (like *.hk* for Chinese or *.ca* for French) exists, or when the other language is much lower density, the actual number of automatically detectable parallel pages is considerably smaller: for example, (Resnik & Smith 2003) find less than 2,000 English-Arabic parallel pages for a total of 2.3m words.

For medium density languages parallel web pages turn out to be a surprisingly minor source of parallel texts. Even in cases where the population and the economy is sizable, and a significant monolingual corpus can be collected by crawling, mechanically detectable parallel or bilingual web pages exist only in surprisingly small numbers. For example our 1.5 billion word corpus of Hungarian (Halácsy et al. 2004) with 3.5 million unique pages, yielded only 270,000 words (535 pages), and a 200m word corpus of Slovenian (202,000 pages) yielded only 13,000 words (42 pages) using URL parallelism as the primary matching criterion (Chen & Nie 2000). No doubt the numbers achieved by this pilot project could be improved with larger crawls and more refined URL matching, but they are not promising when one intends to employ this approach for the corpus building task for medium density languages.

A recent successful effort to employ the web-based method to build a large-scale French-English corpus illustrates our point (*Giga-FrEn*, unpublished): this corpus is based on crawling the Canadian, French and EU top level domains, and finding matching URLs. The effort resulted in a parallel corpus of approximately 1.25b tokens in 22.5m bisentences. Of these bisentences 64,000 contained the word *France*, and 1,872,000 contained the word *Canada*, clearly showing the bias towards the bilingual domain.

Therefore, one needs to resort to other sources, many of them impossible to find by mechanical URL comparison, and often not even accessible without going through dedicated query interfaces. We discuss the nature of these resources using Hungarian as our primary example.

**Literary texts** The Hungarian National Library maintains a large public domain digital archive ‘Hungarian Electronic Library’ (<http://mek.oszk.hu/indexeng.phtml>) with many classical texts. Comparison with the Project Gutenberg archives at <http://gutenberg.org> yielded well over a hundred parallel texts by authors ranging from Jane Austen to Tolstoy. Equally importantly, many works still under copyright were collected from online book collections. While we can not publish most of these texts in either language, we publish the aligned sentence pairs alphabetically sorted. This “shuffling” somewhat limits usability inasmuch as higher than sentence-level text layout becomes inaccessible, but at the same time makes it prohibitively hard to reconstruct the original texts and contravene the copyright. Since shuffling nips copyright issues in the bud, it simplifies the complex task of disseminating aligned corpora considerably. For version 2.0 of the corpus, the size of the modern literature subcorpus was increased approximately fourfold, from 450k bisentences to 1670k bisentences.

**International Law** From the Universal Declaration of Human Rights to the Geneva Convention many important legal documents have been translated to hundreds of languages and dialects. Those working on the languages of the European Union have long availed themselves of the CELEX database. For version 2.0 of the corpus, we added CELEX material that became available since the creation of the original corpus. This increased the size of the subcorpus by ~40%.

**Movie captioning** The thriving semi-underground community of amateur subtitle translators often release subtitles even before a picture is distributed in their country. To the best of our knowledge our research group was the first to exploit this valuable data as a

source of parallel text. (More recently, the OPUS project (Tiedemann 2009) created a large multilingual parallel corpus based on such data.) We provide this subcorpus after shuffling, just like the modern literature subcorpus.

**Software internationalization** Multilingual software documentation is increasingly becoming available, particularly for open source packages such as KDE, Gnome, OpenOffice, Mozilla, the GNU tools, etc (Tiedemann & Nygaard 2004).

**Bilingual magazines** Both frequent flyer magazines and national business magazines are often published with English articles in parallel. Many magazines from Scientific American to National Geographic have editions in other languages, and in many countries there exist magazines with complete mirror translations (for instance, *Diplomacy and Trade Magazine* publishes every article both in Hungarian and English).

There is no denying that the identification of such resources, negotiating for their release, downloading, format conversion, and character-set normalization remain labor-intensive steps, with good opportunities for automation only at the final stages. But such an effort leverages exactly the strengths of medium density languages: the existence of a joint cultural heritage, of national institutions dedicated to the preservation and fostering of culture, of multinational movements (particularly open source) and multinational corporations with a notable national presence, and of a rising tide of global business and cultural practices. Altogether, the effort pays off by yielding a corpus that is orders of magnitude larger, and covering a much wider range of jargons, styles, and genres, than what could be expected from parallel web pages alone. Table 7.4 summarizes the different types of texts and their sizes in Hunglish Corpus version 2.0. Size and token number are to be interpreted as sum of values for Hungarian and English.

source	docs	size (MB)	tokens (M)	bisentences (k)
Modern literature	278	217	37.1	1670
Classical literature	83	100	17.2	652
Movie subtitles	437	19	3.2	343
Software docs	9	9	1.2	135
Legal text	20378	399	56.6	1351
<b>Total</b>	21185	744	115.3	4151

Table 7.4: Distribution of text types in Hunglish Corpus version 2.0

### 7.4.2 Preparing the corpus

After some elementary format-detection and conversion routines such as `catdoc` and `pdftotext` which are standard in the open source world, we have a corpus of raw text consisting of assumed parallel documents. While the texts themselves were collected and converted predominantly manually, the aligned bicorpus is derived by entirely automatic methods.

After the texts are in raw format, automatic language- and character encoding detection makes sure that the bitexts are correct. For manually collected text these steps are nothing more than a sanity check, but as we will see in the next chapter, the same pipeline is incorporated into our online bitext query service, where document pairs uploaded by our anonymous end users can not be expected to be always well-formed and matching.

The next steps in the pipeline are sentence and paragraph boundary detection and word tokenization, performed by our `huntoken` tool. `huntoken` has rule-sets available for both Hungarian and English. After this stage, bitexts are dropped if sentence- or character counts differ too much between languages.

For languages with more complex morphology such as Hungarian, it makes sense to conflate by stemming morphological variants of a lexeme before the texts are passed to the aligner. We used `hunmorph` both for Hungarian and English. (For performance reasons, we

simply choose the shortest possible stem instead of doing a full morphological disambiguation by `hundisambig`. This does not seem to affect accuracy measurably.)

The most important ingredient of the pipeline is of course automatic sentence alignment which we carried out using the `hunalign` tool, described in detail in the previous sections. For the Hungarian-English language pair we could rely on our large bilingual lexicon, but as we noted, `hunalign` can work without a lexicon resource, with only a minimal hit in performance. After the alignment ladders are created, they are converted to text based on the untokenized, unstemmed data.

The data is provided in two simple formats: bisentence files consist of tab-separated, matching sentence pairs. For this, low quality bisentences are thrown away based on various simple heuristics such as: too few alphabetic characters. Too long repeating pattern. Some patterns that are characteristic of OCR problems in subtitle text. Because of quality filtering, the full reconstruction of the raw texts is impossible from the bisentence files. Where copyright considerations made it necessary, the lines of bisentences files were shuffled (sorted alphabetically). When copyright is not an issue, we also provide complete alignments that preserve the whole of both input texts with ordering, including those segments that were not in one-to-one correspondence.

In the past ten years, much has been written on bringing modern language technology to bear on low density languages. At the same time, the bulk of commercial research and product development, understandably, concentrated on high density languages. To a surprising extent this left the medium density languages, spoken by over half of humanity, under-researched. In (Varga et al. 2005) we attempted to address this issue by proposing a methodology that does not shy away from manual labor as far as the data collection step is concerned. We found that in the case of Hungarian, harvesting web pages and automatically detecting parallels turns out to yield only a meager slice of the available data. Instead, we proposed several other sources of parallel texts based on our experience with creating a 115 million word Hungarian-English parallel corpus.

Our solution to issues regarding copyright was ‘shuffling’ the bisentences of the affected subcorpora. This approach was vindicated by the fact that the Hunglish Corpus was published by the Linguistic Data Consortium as part of their Corpus Catalog. (<http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2008T01>)

## 7.5 Resource built - the JRC-Acquis Corpus

This Section describes the JRC-Acquis Corpus (Steinberger et al. 2006), available at <http://langtech.jrc.ec.europa.eu/JRC-Acquis.html>. JRC-Acquis is a large multilingual parallel text corpus built by an international team of language technologists. The author contributed to the creation of the parallel corpus by adapting and applying the **hunalign** technology to build sentence alignments for all possible language pairs.

The Corpus is one of the most important resources of multilingual natural language processing. Its main use is probably as training data for statistical machine translation systems such as (Turchi et al. 2009), but it was used in experiments in many fields such as cross-language information retrieval (Talvensaari 2008), (Potthast et al. 2008), multilingual sentiment analysis (Bautin et al. 2008), and cross-language plagiarism detection (Potthast et al. 2011).

JRC-Acquis is created from the so-called Acquis Communautaire, the total body of European Union law applicable in the the EU Member States. The Acquis Communautaire is officially translated to 22 languages: Bulgarian, Czech, Danish, German, Greek, English, Spanish, Estonian, Finnish, French, Hungarian, Italian, Lithuanian, Latvian, Maltese, Dutch, Polish, Portuguese, Romanian, Slovak, Slovene and Swedish. The JRC-Acquis data is extracted from these translations. As seen on Table 7.5, the number of documents in JRC-Acquis varies across languages, as some of the translations are missing.

As detailed above, **hunalign** works in three phases for each document pair it encounters: alignment, dictionary building, realignment. Because of the extreme number of small documents to be processed, we had to slightly adapt the implementation of **hunalign** for

Language	Doc #	Token #	Char #
bg	11384	16140819	104522671
cs	21438	22843279	148972981
da	23624	31459627	213468135
de	23541	32059892	232748675
el	23184	36453749	239583543
en	23545	34588383	210692059
es	23573	38926161	238016756
et	23541	24621625	192700704
fi	23284	24883012	212178964
fr	23627	39100499	234758290
hu	22801	28602380	213804614
it	23472	35764670	230677013
lt	23379	26937773	199438258
lv	22906	27592514	196452051
mt	10545	20926909	128906748
nl	23564	35265161	231963539
pl	23478	29713003	214464026
pt	23505	37221668	227499418
ro	19211	30832212	182631277
sk	21943	26792637	179920434
sl	20642	27702305	178651767
sv	20243	29433037	199004401
Total	463,792	636,216,050	4,288,962,348

Table 7.5: Corpus sizes for JRC-Acquis.

the task at hand, without deviating from this basic outline. For a fixed choice of language pair, the modified implementation runs in three phases. First, it builds alignments for each document pair. The one-to-one segments found in this first round of alignment are randomly sampled (10,000 sentence pairs in the case of the Acquis corpus) to feed the automatic lexicon-building. In the third phase alignment is re-run, this time also considering similarity information based on the automatically constructed bilingual lexicon, one for each language pair.

We already noted that number tokens are treated specially by **hunalign**: similarity of the sets of number tokens in the two sentences is considered. This special treatment is especially useful for legal texts: in the Acquis corpus, 6.5 percent of the tokens are numbers.

We mention that the original authors of JRC-Acquis processed the corpus using the so-called **Vanilla** aligner. **Vanilla** is an implementation of the Gale-Church algorithm (Gale & Church 1991), thus it only considers sentence length information. The author of the thesis joined the collaboration after the need arose to process the data with a more modern approach. The original **Vanilla** alignments are also provided with the data. In a manual evaluation of a small sample subcorpus, (Kaalep & Veski 2007) finds that the **hunalign** suggestion is always the correct one when the **hunalign** and **Vanilla** alignments differ. Unlike **Vanilla**, **hunalign** does not emit 2-2 segments, but it can deal with the splitting of a sentence into more than two sentences.

For a given language pair and the whole JRC-Acquis Corpus, the running time of each of the three phases of the **hunalign** algorithm was about ten minutes on a fast personal computer. This made it perfectly feasible to run the algorithm on all 231 language pairs of the corpus. We note that after incremental changes to the corpus, it was not necessary to re-run the first two phases.

## 7.6 Later work

This section reviews some of the results reported since the release of **hunalign**.



(Abdul Rauf et al. 2012) compares the performance of sentence alignment systems by training statistical machine translation (SMT) systems on their output, and evaluating the performance of the resulting SMT systems according to three popular evaluation metrics. For each of the three evaluations of the French-English SMT system, **hunalign** was ‘statistically tied for first place’, that is, its score was within one standard deviation of the best score.

(Yu et al. 2012) reports on a sentence alignment system tuned for literary works. They use **hunalign** with default settings (without the dictionary building and realignment phases) as a baseline, and report a significant improvement in precision on their corpora.

(Tóth et al. 2008) created a Hungarian-English sentence aligner system by incorporating into the similarity score the output of a named entity recognition system for both languages. They tuned and evaluated the system on their own gold standard Hungarian-English parallel corpus. The system achieved significantly higher precision than **hunalign** (93.41% versus 89.93%), but mostly at the expense of recall: the F-scores were 93.70% for **hunalign**, and 93.98% for the named entity recognition-based system.

## 7.7 Applications

**hunalign** is widely used by the multilingual natural language processing community. It was used to create four of the largest five public multilingual parallel corpora we are aware of: (i) the aforementioned JRC-Acquis corpus (Steinberger et al. 2006); (ii) OPUS (Tiedemann 2009); (iii) Parasol (Waldenfels 2011); and (iv) InterCorp (Rosen & Vavřín 2012). The exception is (v) EUROPARL (Koehn 2005) which was aligned with the earlier Vanilla aligner, but in subsequent work (Koehn 2010) himself recommends the use of **hunalign**.

**hunalign** was integrated into several software tools as a component. The two most notable are probably **UPlug** and **LF Aligner**.

**UPlug** (Tiedemann 2002) is a framework for parallel text processing. It comes with a graphical user interface to manually correct alignments at various levels. Most notably, **UPlug**

was used to create the OPUS parallel corpus (Tiedemann 2009) relying on its `hunalign` plugin for the task of sentence alignment.

Bisentence databases (translation memories) are important resources of computer-assisted translation systems. Somewhat surprisingly, these expensive software systems rarely incorporate automatic alignment functionality. András Farkas created **LF Aligner** <http://sourceforge.net/projects/aligner/>, a convenient GUI wrapper around `hunalign` and `partialAlign`, tailored to professional translators who would like to create their own translation memories without having to learn script programming. It is used by thousands of translators worldwide. **LF Aligner** supports several common document formats as input, and several common translation memory formats as output. It provides built-in bilingual lexicons for many language pairs to increase alignment accuracy. **LF Aligner** can build multilingual translation memories by repeatedly calling `hunalign` for a pivot language versus all the other languages, and iteratively merging compatible bisentences into the multisentence set.

The Hunglish Corpus has been used as training data for machine translation (Hócsa & Kocsor 2006), and in a word-sense disambiguation experiment (Miháltz & Pohl 2006). It was also employed as part of the training data in the ‘2008 ACL Workshop on Statistical Machine Translation’ (Callison-Burch et al. 2008) and ‘2009 EACL Workshop on Statistical Machine Translation’ <http://www.statmt.org/wmt09/translation-task.html> shared tasks.

The following chapter on bitext querying will introduce a full-scale web-based application for end users that builds on all results of this chapter.

## Chapter 8

### The Hunglish bitext query system

This chapter starts with documenting the `huntools` multilingual text processing framework. `huntools` is a lightweight text processing framework that was created to integrate the various tools presented in previous chapters. We present a list of text processing pipelines we created in the framework.

Next we introduce the Hunglish bitext query service. The goal of this web-based system is two-fold: first, it is a tool for human translators that helps them find translations of phrases in context. Second, thanks to its crowdsourcing functionality, one important ‘side effect’ of this usage is the semi-manual collection and manual evaluation of corpus data. Our system is a full reimplementation of the bitext query system originally created by Péter Halácsy. The main new features of the reimplementation are related to crowdsourcing functionality: the capability to add bidocuments to a running service, and the chance for users to vote on bisentence quality. Duplicate filtering was also added.

This chapter differs from the previous ones in several respects. It does not follow the their structure, focusing on the components of a single end-user application, and it mostly deals with implementation and software design issues rather than algorithmic questions.

The offline component is the result of joint work with Attila Zséder (Recski et al. 2009). The architecture of the software is the design of the present author. The implementation of the framework is the work of Attila Zséder. Plugging the `hun*` tools and other language processing utilities into this framework was the work of the present author. The online component is joint work with Péter Gergő Barna, previously unpublished. The architecture

of the software is the result of joint work, implementation is largely the work of Péter Gergő Barna.

### 8.1 The hunttools multilingual text processing framework

There are several natural language processing toolkits in existence, some of them widely used. Some of the more important examples are the Java-based GATE (General Architecture for Text Engineering) (Cunningham et al. 1997) and the Python-based NLTK (Natural Language Toolkit) (Bird 2006). Two other notable toolkits are the Java-based UIMA (Ferrucci & Lally 2004) and OpenNLP frameworks. All of these systems provide their own standard mechanisms to facilitate communication among their components. When a research group decides to build its own text processing framework instead of using one of the above, this decision requires strong justification. Below we give such a justification, explaining the design principles behind our framework.

Given the Unix background of our research group, we have a large amount of accumulated expertise in writing short text processing tools based on standard Unix tools such as **sed**, **grep**, **awk** and others. We intended to make use of this expertise. This constrained the design of the framework in two important ways:

First, textual data travels in a trivial tabulator-separated text representation. The limitations of this very simple file structure do not interfere with the applications we had in mind. For example, we did not intend to carry information about detailed document structure, formatting, and typesetting. (If we want to add such a feature in the future, stand-off annotation will be the method to achieve this without having to reorganize the pipeline to work with some recursive data representation such as XML.) A large advantage of tabulator-separated data is that character escaping is never required: the new line and the tabulator characters are the only reserved symbols, and they are not allowed inside fields. (Compare this with the Comma-Separated Values file format, which is quite hard to write a complete parser for.)

The second constraint is that the framework has to be able to spawn any kind of sub-processes as components. Contrast this with toolkits like UIMA and GATE, that in practice force the component developer to work in limited list of programming languages. (Java and C++, in the case of UIMA.)

After these considerations, one good candidate for a non-intrusive driver component for our framework was **zymake** (Breck 2008), a framework for organizing experiments that shares the philosophy of the popular Unix tool **make**. We decided against this and similar tools because we had some features in mind (like daemonization, see below) that would have been hard to implement under these. Creating **huntools** took only about a person-week of effort, a fact we feel justifies succumbing to the not-invented-here syndrome.

### 8.1.1 Conventions, implementation details

The input, intermediate results and output of a run are stored in a trivial standardized file structure. The results of step *x* for language *l* for an input with identifier *id* can be found in file *l/x/id.l.x*. (For example, *hu/sen/12.hu.sen* contains sentence-segmented Hungarian data, and *align/qf/12.align.qf* contains quality-filtered bilingual parallel data.) The reason for the redundancy in the full file path is that this makes it possible to move files without losing information contained in the path.

The tasks to be completed for the data are specified in two driver files: the state graph file and the commands file. The state graph file specifies the workflow as a directed acyclic graph of commands. A command takes one or more data files as input, and outputs a previously non-existing data file. The command file describes the Unix process corresponding to commands. An example part of a state graph file:

```
# sentence alignment using hunalign.
# input is tokenized text, output is a ladder file.
# (a ladder is a list of pairs of sentence indices)
hu/tok, en/tok -> align/ladder : align

# use the (non-tokenized) sentences and the ladder
```

```
# to reconstruct bisentences.
align/ladder, hu/filtersen, en/filtersen -> align/text: align2text

# quality filter. throws away low quality bisentences.
align/text -> align/qf : qf
```

An example part of a command file:

```
aligner_dic=%(resources_dir)s/hunalign/hu-en.stem.dic

align_cmd: %(binaries_dir)s/hunalign %(aligner_dic)s

qf_cmd: bash %(scripts_dir)s/qualityfilter.sh |\
python %(scripts_dir)s/enoughalpha.py
```

By convention, if there is one input file, then the Unix command is executed as a pipe mapping input to output. If there are several input files, they will be appended as the last command line arguments of the Unix command.

The core of this very lightweight framework that interprets driver files and spawns and logs Unix processes in the required order is just about 200 lines of Python code. A minor additional source of complexity is a set of extra features that make large batch runs significantly faster:

- **Parallelization:** The system is capable of harnessing the power of a computer cluster. The members of the cluster must mount the same file system. The basic unit of parallelization is the document. That is, for a single document, only one machine of the cluster receives load. This limitation greatly simplifies the framework, but is only a minor slowing factor, because the state graph is typically very sequential in nature, so the processing of a single document is an inherently sequential task.
- **Daemonization:** The supervised, machine-learning based taggers and chunkers we employ usually load a large model file at startup. (Sometimes this is larger than 100MB, as in the case of some **hunner** versions.) If we run such a tagger as a pipe for many

small data files separately, then most of the runtime will consist of the same initialization step again and again. To avoid this, we can run such a tool in a so-called daemon mode, where it starts up only once, and then waits for data to arrive on Unix sockets. The framework hides from the user most of the necessary administration, making it easy to set up tools in this fashion. Note that this functionality is also required when using the tool as part of some (typically web-based) service.

### 8.1.2 Pipelines

During our work, our research team often builds pipelines in the form of driver files for the framework. The three most notable of these is briefly described below. It is trivial to chain these one after another into a single large pipeline working from native document formats to factored Statistical Machine Translation models. But this is seldom useful for research tasks, and more often they are used individually.

The *hunglish* pipeline starts from bilingual document pairs in standard document formats, and extracts bisentences from them. The pipeline was used to create Version 2.0 of the Hunglish Corpus. (Version 1.0 was created with a more or less ad hoc collection of scripts.) As we will see in the next section, the pipeline is also a part of the Hunglish web-based application, where it is responsible for the capability to expand the bisentence database with data from uploaded document pairs. The pipeline consists of the following tasks:

- raw conversion: Turn doc, pdf, html, srt (subtitles), txt (unformatted ISO Latin- and Unicode text) into unformatted Unicode text.
- sen: Sentence-segment and paragraph-segment text.
- meta: Collect metadata (language detection, encoding detection, character- and sentence counts).

- `filtersen`: Based on metadata, drop data if it is incompatible with target. (Say, it is not in the required language, or there is a discrepancy between sentence- or character counts.)
- `tok`: Tokenize text.
- `stem`: Stem text.
- `align`: Align two tokenized and stemmed document pairs. The result is a ‘ladder’ representing correspondences between sentence ids.
- `ladder to text`: Collect the sentence pairs for the non-stemmed, non-tokenized text, based on the ladder.
- `quality filter`: Throw away low quality bisentences based on various simple heuristics.
- `bimeta`: Collect metadata about alignment quality.

The *analysis* pipeline starts with a single-language tokenized document, and adds morphological, named entity and NP chunk information. It is in principle language-independent, but requires language resources for morphology and for supervised learning, so currently it can only be used for Hungarian and English language data. It can be chained after the *hunglish* pipeline if required.

- `morph`: Morphological analysis.
- `pos`: Morphological tagging.
- `disambig`: Morphological disambiguation.
- `ner`: Named entity recognition.
- `chunk`: noun phrase chunking.



The third, *giza* pipeline uses the GIZA++ translation model builder (Och & Ney 2003) and the Moses Statistical Machine Translation framework (Koehn et al. 2007) to build word alignments, SMT models and automatically acquired dictionaries. It can be chained after the analysis pipeline, so it can exploit morphology and tags provided by the former pipeline, using factored language models.

- to giza: Converting parallel, tagged text to the standard file format of GIZA++.
- model: Build an IBM Model 5 from the data.
- word align: Build a word alignment with the model.
- mooses: Use the translation and language models for actual translation of monolingual text.
- bi-np: Heuristically build an NP-chunk alignment based on the GIZA++ model and the NP chunkings.

For historical reasons, the *analysis* and *giza* pipelines have an important limitation: they work on one-byte Latin (ISO-8859-1 and ISO-8859-2) encoded text rather than UTF-8. This limitation is inherited from *ocamorph*. The *hunglish* pipeline is Unicode-based, or more exactly, stemming is the only place where it does character conversion. Note that the stemmed text is discarded after the alignment ladder is found, so the pipeline does not drop UTF-8 characters during processing.

## 8.2 The Hunglish bitext query system

The Hunglish bitext query system is a web-based service that makes it possible for users to search in a large collection of bisentences. Its intended audience is translators, helping them find translations of phrases in context. The system is augmented with crowdsourcing functionality: first, users can vote on the accuracy of translations, thus improving the quality

of the parallel corpus.<sup>1</sup> Second, users can upload their own document pairs from which the system extracts a set of bisentences. Taken together, these two features let us build a semi-manually collected and manually evaluated bisentence corpus.

The system consists of two major components, an offline (asynchronous) and an online (synchronous) component. The offline component takes a pair of documents as input, extracts a set of bisentences from them, and adds this set to an index. The core of the offline component is the *hunglish* pipeline of the *hunttools* framework. The online component answers user search queries on the index of bisentences. The two systems are integrated into a web-based application that is capable of answering search requests even while simultaneously adding new user-supplied bitexts to its database. Downvoted bisentences almost immediately drop to a lower position on the result list.

### 8.2.1 Architecture

The design of the bitext system is based on the producer-consumer design architecture. Below we describe the way our application employs this pattern. The abstract description is easier to follow if we keep in mind that in the application, there are two kinds of work items: bidocuments and bisentences. Typical examples of workers are duplicate filtering and indexing.

The producer-consumer design is based on the notions of workers and work items. At all times, items have a unique state that describes which one of the workers (if any) needs to process them. Each worker can poll the database for items that it should work on, at any given time. If such a query gives a nonzero number of results, the worker “grabs” some or all of the items, and processes them. As a result, it might modify the items in the database, and change their state to processed (possibly unsuccessfully processed). Such a processed item can become input for a next worker. Some of the actions have side effects. (A typical example

---

<sup>1</sup>Translations can be faulty for several reasons: errors made by the sentence aligner, errors made by the human translator, or translations that are correct in a broader context but incorrect in isolation.

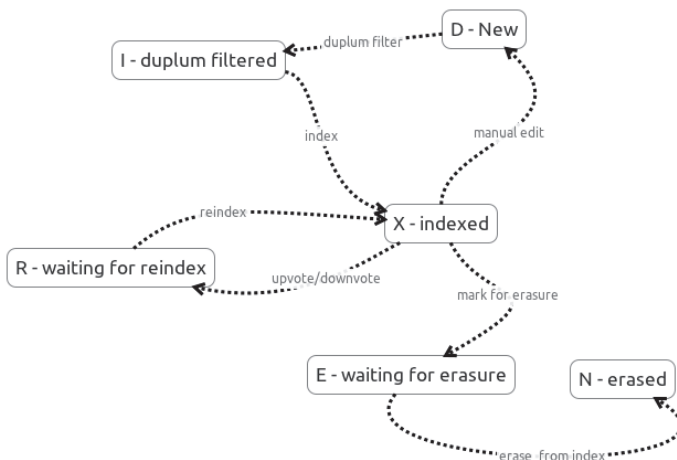


Figure 8.1: State transition graph of bisentences.

of a side effect is the addition of a bisentence to the index. In this case, the only non-side effect of the corresponding action is the changing of the bisentence state from waiting-for-index to added-to-index.)

The relationships between workers and items can be described by the state graph on Figure 8.1. An important property of our state graph is that the large scale functionality of the system does not depend on the order in which the workers grab the work items. Processes denoted by the arcs of manual edit, upvote/downvote and mark for erasure are triggered by the users. Duplicate filter, index, reindex, erase from index on the other hand are all scheduled to run regularly. (They are also triggered by other processes, but that is only for the purpose of making the application more responsive to user input.)

For each of the workers, we can tune the frequency of waking up and polling for new input, and tune the amount of items it is allowed to grab at once. The optimal settings depend on performance- and user experience considerations.

### 8.2.2 Indexing and retrieval

After the `hunttools` pipeline transforms an uploaded document into a set of bisentences, these bisentences enter into the database of work items. As the state transition graph on Figure 8.1 shows, the first step is duplicate detection: bisentences that are already part of the index should not be added for a second time. Duplicate detection is based on a hash of bisentences stripped of punctuation.

We implemented two different duplicate filtering algorithms with different scaling properties. Both of them are working components of the system. For a small amount of newly arrived bisentences, their hashes are individually checked against the hash field of the table of bisentences, leading to a slow algorithm with zero overhead. For a large batch of new data, we sort the full table of (old and new) bisentences by hash, and mark newly arrived duplicates during a linear pass through the sorted data. The run-time of this algorithm does not directly depend on the amount of new data, but scales by  $O(n \log n)$  in the amount of old *and* new data. (For our current setup, adding even a single bisentence by this ‘large batch’ method would take 6 minutes, compared to the 7ms for the ‘small batch’ method.)

For text indexing and retrieval we rely on the Lucene information retrieval engine (Hatcher et al. 2010). Lucene supports ranking retrieval results using a vector-space model (Manning & Schütze 1999).

Indexed bisentences and search queries both pass through a lemmatization phase. This functionality is provided by `jmorph`, which can work with any `hunspell` language resource, so lemmatization can be implemented for a large number of languages. Unlemmatized text is also preserved, and exact matches are preferred over lemma matches.

One feature of our system that is inherited from the underlying Lucene feature set is that we can ‘boost’ bisentences, that is, we can specify certain that bisentences deserve a higher (or lower) position on the query result list than others. This is used to implement the voting functionality. A second use of this feature in our system is that bisentences from documents uploaded by operators get more prominent positions on query result lists than bisentences from documents uploaded by end users.

### 8.2.3 The Hunglish query interface

Users of the Hunglish bitext query system can set any number of space-separated query phrases on the source or target language side, or even both sides at once. The queries are translated by the system to complex Lucene queries that refer to one or more of four fields: the lemmatized/unlemmatized source/target language fields.

The syntax of these query phrases is easiest to describe through a series of examples:

- The simplest example of a query phrases is a word: the Hungarian side *ellopták* query results in bisentences with this Hungarian word in their Hungarian sentence. Other inflected forms (*ellopnám*, *ellopott*) are also returned as results, but are ranked lower than the exactly matching version.
- To prohibit results with other inflected forms, the term should be surrounded with the `< >` parentheses. So `<ellopták>` will not give results with only, say, *ellopott* in them. This is called an exact match query.
- One can search for multi-word search terms. The three equivalent ways of doing this are:
  - Quotation: `"back to normal"`
  - Dashes: `back-to-normal`
  - Dots: `back.to.normal`

hunglish.hu/search?huSentence=&enSentence=stroke+of+ Google

Hungarian: English: stroke of luck in genre: All Search

Results 1 - 20 of 576892; Page 1 of 28845; Pages: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50









































Dictionary hirtelen nagy szerencse	stroke of good luck	 
Dictionary jó szerencse	stroke of good luck	 
Dictionary váratlan nagy szerencse	stroke of good luck	 
Dictionary szerencse	stroke of luck	 
Dictionary váratlan szerencse	stroke of luck	 
Literature Micsoda szerencsével	What a stroke of luck, indeed.	 
Literature És egy év elteltével a szerencse úgy hozta, hogy valóban kaptam.	And sure enough, after about a year, by a stroke of luck it happened.	 
Literature Hihetetlen szerencséjének köszönheti, hogy felfedezte.	Only a stroke of luck had given it away.	 
Literature Itt senki sem támadhatja meg s azonfelül szerencséje is volt.	Certainly no one could attack him here--and moreover he had a stroke of luck.	 
Literature Az öreg landoló megtalálása tényleg váratlan szerencse volt.	Finding the old lander had indeed been a stroke of luck.	 
Subtitles Az volt a szerencsém, hogy nincs védőkorlát az ablakokon, szóval...	The one stroke of luck is that there's no safety bars on the windows, so...	 
Literature Cathcart ezredes a jó szerencséjének éppen ilyen csapásáért imádkozott, mint Duluth őrnagy haláláért.	Colonel Cathcart had been praying for just some stroke of good luck like Major Duluth's death.	 
Literature A férfi csak állt és bámult, képtelen volt elhinni, hogy ilyen szerencse érte.	He just stood and stared, unable to believe what had happened, what a stroke of luck.	 
Literature De ez sem sikerült volna, ha ismét nem játszik kezükre a szerencse, amely gyakorta bár nem mindig jutalmazza azokat, akik megérdemlik.	It would never have been possible at all without one of those strokes of luck that sometimes - not always -favour those who deserve them.	 
Literature A férfi, akinek csupán egyetlen fia van, ki van szolgáltatva a balszerencse csapásainak - mondta Uther.	The man with only one son walks always at the mercy of any stroke of bad luck," said Uther.	 
Literature Csak hogy megünnepeljék a szerencséjét az emberek mégsem minden nap pottyán az ólébe tiz öt font.	Just to celebrate his stroke of luck; after all, it isn't every day that ten pounds - five pounds - drops out of the sky into your lap.	 
Literature Azt mondom, a szerencse hozta úgy, de az igazat megvalva, úgyis talpra álltam volna.	I say by a stroke of luck, but the fact is that I was bound to fall on my feet.	 
Literature Abban is rájuk mosolygott a szerencse, hogy Aaront feltétel nélkül befogadta a család, és automatikusan meghívták minden összejövetelre.	Another good stroke of luck was that Aaron had been completely accepted by the family, and was now routinely included in every gathering.	 
Dictionary előnyös üzlet	good stroke of business	 
Dictionary sikeres üzlet	good stroke of business	 

Figure 8.2: A screenshot from the Hunglish bitext query system.

- One can mix the exact match syntax with multi-word search terms, but mixing parentheses and quotation signs (<"*back to normal*">) is not allowed. So the exact match versions of the previous examples would look like this: <*back to normal*>, <*back-to-normal*>, <*back.to.normal*>.
- Any query phrase is allowed to have one of the following two modifiers as prefixes. Space is not allowed between the modifier and its modifyee.
  - The Prohibited modifier (minus sign) negates the query phrase. So, for example Hungarian: *fél* English: *-scared* returns bisentences where there is a word in the Hungarian sentence which stems the same as "*fél*", but there is no word in the English sentence which stems the same as "*scared*". The Prohibited modifier can be combined with exact match syntax and multi-word search terms.
  - The Required modifier (plus sign) requires that the term after the '+' symbol exist somewhere in the sentence. Without any modifier the phrase is optional. So, for example English *buy into* return bisentences where the words *buy* OR *into* but the query *+buy +into* returns only bisentences with the two words in it.

### 8.2.4 Usage

We set up a webservice running our system for the Hungarian-English language pair at <http://hunglish.hu>. We added a subset of the Hunglish Corpus,<sup>2</sup> a total of 2.4 million bisentences in 1000 documents. We also added the approximately 250,000 items of the Vonyó bilingual lexicon. We use the aforementioned boosting functionality to make these items more visible than regular bisentences.

We note that the *hunglish* pipeline has a variation that is stripped of all language-dependence without any major impact on accuracy, and the *jmorph* lemmatizer can use any *hunspell* resource. This means that setting the service up for some specific language

---

<sup>2</sup>The top 1000 bidocuments of the Corpus, when ordered by bisentence count.

pair is simple task. We verified this claim by setting a test system up for the Greek-Slovak language pair based on JRC-Acquis data.

The service, still in beta at the time of writing, and without a polished user interface, already achieved significant web presence. It serves 150,000 search queries monthly to 10,000 visitors. Visitors reported 2,000 erroneous alignments in the Hunglish Corpus, and uploaded 37 new document pairs yielding 17,000 new parallel sentences.



## Bibliography

- Abdul Rauf, S., Fishel, M., Lambert, P., Noubours, S. & Sennrich, R. (2012), Extrinsic evaluation of sentence alignment systems, *in* ‘LREC Workshop on Creating Cross-language Resources for Disconnected Languages and Styles (CREDISLAS)’, Istanbul (Turkey).
- Abney, S. P. (1991), Parsing by chunks, *in* ‘Principle-Based Parsing’, Kluwer Academic Publishers, pp. 257–278.
- Bautin, M., Vijayarenu, L. & Skiena, S. (2008), International sentiment analysis for news and blogs, *in* ‘Proceedings of the International Conference on Weblogs and Social Media (ICWSM)’.
- Bird, S. (2006), NLTK: the natural language toolkit, *in* ‘Proceedings of the COLING/ACL on Interactive presentation sessions’, COLING-ACL ’06, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 69–72.
- Bishop, C. M. (2007), *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer.
- Borthwick, A. E. (1999), A maximum entropy approach to named entity recognition, PhD thesis, New York University, New York, NY, USA. AAI9945252.
- Borthwick, A., Sterling, J., Agichtein, E. & Grishman, R. (1998), NYU: Description of the MENE named entity system as used in MUC-7, *in* ‘Proceedings of the Seventh Message Understanding Conference (MUC-7)’.
- Bottou, L. (2003), Stochastic Learning, *in* ‘Advanced Lectures on Machine Learning’03’, pp. 146–168.

- Bourigault, D. (1992), Surface grammatical analysis for the extraction of terminological noun phrases, *in* 'Proceedings of the 14th conference on Computational linguistics - Volume 3', COLING '92, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 977–981.
- Brants, T. (2000), TnT – a statistical part-of-speech tagger, *in* 'Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)', Seattle, WA.
- Breck, E. (2008), zymake: a computational workflow system for machine learning and natural language processing, *in* 'Software Engineering, Testing, and Quality Assurance for Natural Language Processing', SETQA-NLP '08, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 5–13.
- Brown, P. F., Pietra, V. J. D., Pietra, S. A. D. & Mercer, R. L. (1993), 'The mathematics of statistical machine translation: parameter estimation', *Computational Linguistics* **19**(2), 263–311.
- Brown, P., Lai, J. & Mercer, R. (1991), Aligning sentences in parallel corpora, *in* 'Proceedings of ACL29', pp. 169–176.
- Burges, C. J. C. (1998), 'A Tutorial on Support Vector Machines for Pattern Recognition', *Data Min. Knowl. Discov.* **2**(2), 121–167.
- Callison-Burch, C., Fordyce, C., Koehn, P., Monz, C. & Schroeder, J. (2008), Further meta-evaluation of machine translation, *in* 'Proceedings of the Third Workshop on Statistical Machine Translation', StatMT '08, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 70–106.
- Chen, J. & Nie, J.-Y. (2000), Automatic construction of parallel english-chinese corpus for cross-language information retrieval, *in* 'Proceedings of the sixth conference on Applied natural language processing', Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 21–28.

- Chen, S. F. (1993), Aligning sentences in bilingual corpora using lexical information, *in* 'Proceedings of the 31st conference on Association for Computational Linguistics', Association for Computational Linguistics, Morristown, NJ, USA, pp. 9–16.
- Chieu, H. L. & Ng, H. T. (2003), Named entity recognition with a maximum entropy approach, *in* 'Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4', CONLL '03, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 160–163.
- Chinchor, N. A. (1998), Proceedings of the Seventh Message Understanding Conference (MUC-7) named entity task definition, *in* 'Proceedings of the Seventh Message Understanding Conference (MUC-7)', Fairfax, VA, p. 21 pages. version 3.5, [http://www.itl.nist.gov/iaui/894.02/related\\_projects/muc/](http://www.itl.nist.gov/iaui/894.02/related_projects/muc/).
- Cho, J. & Garcia-Molina, H. (2000), The evolution of the web and implications for an incremental crawler, *in* 'VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases', Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 200–209.
- Church, K. W. (1988), A stochastic parts program and noun phrase parser for unrestricted text, *in* 'Proceedings of the second conference on Applied natural language processing', Association for Computational Linguistics, Morristown, NJ, USA, pp. 136–143.
- Creutz, M., Lagus, K., Lindén, K. & Virpioja, S. (2005), Morpheme segmentation for highly inflecting and compounding languages, *in* 'In Proceedings of the Second Baltic Conference on Human Language Technologies', pp. 107–112.
- Crystal, D. (1997), *A Dictionary of Linguistics and Phonetics*, 4th edn, Blackwell, Oxford, UK.

- Csendes, D., Csirik, J. & Gyimóthy, T. (2004), The Szeged Corpus: A POS tagged and syntactically annotated Hungarian natural language corpus, *in* 'Text, Speech and Dialogue: 7th International Conference, TSD', pp. 41–47.
- Csendes, D., Csirik, J., Gyimóthy, T. & Kocsor, A. (2005), The Szeged Treebank, *in* 'Proceedings of the 8th international conference on Text, Speech and Dialogue', TSD'05, Springer-Verlag, Berlin, Heidelberg, pp. 123–131.
- Cunningham, H., Humphreys, K., Gaizauskas, R. J. & Wilks, Y. (1997), GATE - a general architecture for text engineering, *in* 'ANLP', pp. 29–30.
- Dimitrova, L., Erjavec, T., Ide, N., Kaalep, H. J., Petkevic, V. & Tufiş, D. (1998), Multext-east: Parallel and comparable corpora and lexicons for six central and eastern european languages, *in* C. Boitet & P. Whitelock, eds, 'Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics', Morgan Kaufmann Publishers, San Francisco, California, pp. 315–319.
- Elekfi, L. (1994), *Magyar ragozási szótár*, MTA Nyelvtudományi Intézet, Budapest.
- Erjavec, T. & Džeroski, S. (2004), 'Machine learning of morphosyntactic structure: Lemmatizing unknown Slovene words.', *Applied Artificial Intelligence* **18**(1), 17–41.
- Erjavec, T. & Monachini, M. (1997), Specifications and notation for lexicon encoding, Technical report, Copernicus Project 106 MULTEXT-East.
- Farkas, R., Simon, E., Szarvas, G. & Varga, D. (2007), GYDER: Maxent metonymy resolution, *in* 'Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)', Association for Computational Linguistics, Prague, Czech Republic, pp. 161–164.

- Farkas, R., Szeredi, D., Varga, D. & Vincze, V. (2010), MSD-KR harmonizáció a Szeged Treebank 2.5-ben, in 'VII. Magyar Számítógépes Nyelvészeti Konferencia', pp. 349–353.
- Fazekas, J., Németh, K., Varga, D., Várhelyi, K. & Pléh, C. (2012), Entropy measures and predictive recognition as mirrored in gating and lexical decision over multimorphemic Hungarian noun forms, in 'Proceedings of the 15th International Morphology Meeting', Vienna, Austria.
- Ferrucci, D. & Lally, A. (2004), 'UIMA: an architectural approach to unstructured information processing in the corporate research environment', *Nat. Lang. Eng.* **10**(3-4), 327–348.
- Fuschetto, A., Tamperi, F., Simi, M. & Vecchi, E. M. (2009), Experiments in tagger combination: arbitrating, guessing, correcting, suggesting, in 'Proc. of Workshop Evalita 2009'.
- Gale, W. A. & Church, K. W. (1991), A program for aligning sentences in bilingual corpora, in 'Meeting of the Association for Computational Linguistics', pp. 177–184.
- Gee, J. P. & Grosjean, F. (1983), 'Performance structures: A psycholinguistic and linguistic appraisal', *Cognitive Psychology* **15**(4), 411 – 458.
- Grosjean, F. (1980), 'Spoken word-recognition processes and the gating paradigm', *Perception and Psychophysics* **28**, 267–283.
- Hajič, J., Krbec, P., Oliva, K., Květoň, P. & Petkevič, V. (2001), Serial combination of rules and statistics: A case study in Czech tagging, in 'Proceedings of the 39th Association of Computational Linguistics Conference', Toulouse, France, pp. 260–267.
- Hakkani-Tür, D. Z., Oflazer, K. & Tür, G. (2000), Statistical morphological disambiguation for agglutinative languages, in 'Proceedings of the 18th conference on Computational

- linguistics', Association for Computational Linguistics, Morristown, NJ, USA, pp. 285–291.
- Halácsy, P. (2006), 'Benefits of deep NLP-based lemmatization for information retrieval', Working Notes for the CLEF 2006 Workshop.
- Halácsy, P., Kornai, A., Németh, L., Rung, A., Szakadát, I. & Trón, V. (2004), Creating open language resources for Hungarian, in 'Proceedings of Language Resources and Evaluation Conference (LREC04)', European Language Resources Association.
- Halácsy, P., Kornai, A., Németh, L., Sass, B., Varga, D., Váradi, T. & Vonyó, A. (2005), A Hunglish korpusz és szótár, in 'III. Magyar Számítógépes Nyelvészeti Konferencia. Szeged'.
- Halácsy, P., Kornai, A., Németh, P. & Varga, D. (2008), Parallel creation of gigaword corpora for medium density languages - an interim report, in 'Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)', European Language Resources Association (ELRA), Marrakech, Morocco.
- Halácsy, P., Kornai, A. & Oravecz, C. (2007), Hunpos: an open source trigram tagger, in 'Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions', ACL '07, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 209–212.
- Halácsy, P., Kornai, A. & Varga, D. (2005), Morfológiai egyértelműsítés maximum entrópia módszerrel (morphological disambiguation with the maxent method), in 'Proc. 3rd Hungarian Computational Linguistics Conf.', Szegedi Tudományegyetem.
- Hatcher, E., Gospodnetic, O. & McCandless, M. (2010), *Lucene in Action*, 2nd revised edition edn, Manning.
- Hócz, A. (2004), 'Noun phrase recognition with tree patterns', *Acta Cybern.* **16**(4), 611–623.

- Hócza, A. & Kocsor, A. (2006), Hungarian-English machine translation using genpar, *in* 'Proceedings of the 9th international conference on Text, Speech and Dialogue', TSD'06, Springer-Verlag, Berlin, Heidelberg, pp. 87–94.
- Horváth, T., Alexin, Z., Gyimóthy, T. & Wrobel, S. (1999), Application of different learning methods to Hungarian part-of-speech tagging., *in* 'ILP', pp. 128–139.
- Jurafsky, D. & Martin, J. H. (2008), *Speech and Language Processing (2nd Edition)* (Prentice Hall Series in Artificial Intelligence), 2 edn, Prentice Hall.
- Kaalep, H.-J. & Veski, K. (2007), Comparing parallel corpora and evaluating their quality, *in* 'Proceedings of MT Summit XI', pp. 275–279.
- Kenyon, J. & Knott, T. (1953), *A pronouncing dictionary of American English*, A Merriam-Webster, Merriam.
- Kilgariff, A. (2007), 'Googleology is bad science', *Comput. Linguist.* **33**(1), 147–151.
- Klein, D., Smarr, J., Nguyen, H. & Manning, C. D. (2003), Named entity recognition with character-level models, *in* 'Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4', CONLL '03, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 180–183.
- Koehn, P. (2005), Europarl: A Parallel Corpus for Statistical Machine Translation, *in* 'Conference Proceedings: the tenth Machine Translation Summit', AAMT, AAMT, Phuket, Thailand, pp. 79–86.
- Koehn, P. (2010), *Statistical Machine Translation*, Cambridge University Press.
- Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A. & Herbst, E. (2007), Moses: open source toolkit for statistical machine translation, *in* 'Proceedings of the 45th

- Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions', ACL '07, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 177–180.
- Kornai, A. (1985), 'The internal structure of Noun Phrases', *Approaches to Hungarian* **1**, 79–92.
- Kornai, A. (1986), 'Szótári adatbázis az akadémiai nagyszámítógépen (A dictionary database of Hungarian)', *Hungarian Academy of Sciences Institute of Linguistics Working Papers* **II**, 65–79.
- Kornai, A. (1989), 'A főnévi csoport egyeztetése [Agreement in noun phrases]', *Általános Nyelvészeti Tanulmányok* pp. 183–211.
- Kornai, A., Halácsy, P., Nagy, V., Oravecz, C., Trón, V. & Varga, D. (2006), Web-based frequency dictionaries for medium density languages, *in* 'Proceedings of the EACL 2006 Workshop on Web as a Corpus'.
- Krynicky, G. (2006), Compilation, Annotation and Alignment of a Polish-English Parallel Corpus, PhD thesis, Poznan University.
- Kuba, A., Bakota, T., Hócz, A. & Oravecz, C. (2003), A magyar nyelv néhány szófaji elemzőjének összevetése, *in* Z. Alexin & D. Csentes, eds, 'Magyar Számítógépes Nyelvészeti Konferencia', Szegedi Tudományegyetem, Informatikai Tanszékcsoport, Szegedi Tudományegyetem, pp. 16–22.
- Kuba, A., Felföldi, L. & Kocsor, A. (2005), Pos tagger combinations on Hungarian text, *in* '2nd International Joint Conference on Natural Language Processing, IJCNLP'.
- Lafferty, J., McCallum, A. & Pereira, F. (2001), Conditional random fields: Probabilistic models for segmenting and labeling sequence data, *in* 'Proceedings of the 18th International Conference on Machine Learning', Morgan Kaufmann, San Francisco, CA, pp. 282–289.



- Lapata, M. & Keller, F. (2004), The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of NLP tasks, *in* S. Dumais, D. Marcu & S. Roukos, eds, ‘HLT-NAACL 2004: Main Proceedings’, Association for Computational Linguistics, Boston, Massachusetts, USA, pp. 121–128.
- Le, Z. (2011), ‘Maximum Entropy Modeling Toolkit for Python and C++’.  
**URL:** [http://homepages.inf.ed.ac.uk/lzhang10/maxent\\_toolkit.html](http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html)
- Lin, C.-J., Weng, R. C. & Keerthi, S. S. (2007), Trust region Newton method for large-scale logistic regression, *in* ‘Proceedings of the 24th International Conference on Machine Learning (ICML)’.
- Lukács, A., Pléh, C. & Racsmány, M. (2007), ‘Spatial language in Williams syndrome: evidence for a special interaction?’, *Journal of Child Language* **34(2)**:311–43.
- MacKay, D. J. C. (2002), *Information Theory, Inference & Learning Algorithms*, Cambridge University Press, New York, NY, USA.
- Magyar, L. & Szentgyörgyi, S. (2011), Vowel zero alternations in Hungarian nominal inflectional and derivational paradigms: An analogy-based statistical approach, *in* ‘4th Syntax, Phonology and Language Analysis Conference, Budapest’.
- Manning, C. & Schütze, H. (1999), *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, MA.
- Markert, K. & Nissim, M. (2007), SemEval-2007 Task 08: Metonymy resolution at SemEval-2007, *in* ‘Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)’, Association for Computational Linguistics, Prague, Czech Republic, pp. 36–41.
- McCallum, A., Freitag, D. & Pereira, F. (2000), Maximum entropy Markov models for information extraction and segmentation, *in* ‘Proceedings of the 17th International Conference on Machine Learning’, Morgan Kaufmann, San Francisco, CA, pp. 591–598.

- Megyesi, B. (2009), The Open Source Tagger HunPoS for Swedish., *in* ‘Proceedings of the 17th Nordic Conference on Computational Linguistics (NODALIDA)’.
- Melamed, I. D. (1998), Empirical methods for exploiting parallel texts, PhD thesis, University of Pennsylvania, Philadelphia, PA, USA. AAI9829948.
- Melamed, I. D. (2000), ‘Models of translational equivalence among words’, *Computational Linguistics* **26**(2), 221–249.
- Mihajlik, P., Fegyő, T., Németh, B., Tüske, Z. & Trón, V. (2007), Towards automatic transcription of large spoken archives in agglutinating languages hungarian ASR for the MALACH project, *in* V. Matoušek & P. Mautner, eds, ‘Text, Speech and Dialogue’, Vol. 4629 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 342–349. 10.1007/978-3-540-74628-7\_45.
- Miháltz, M. (2011), Magyar NP-felismerők összehasonlítása, *in* ‘VIII. Magyar Számítógépes Nyelvészeti Konferencia. Szeged’, pp. 333–335.
- Miháltz, M. & Pohl, G. (2006), Exploiting Parallel Corpora for Supervised Word-Sense Disambiguation in English-Hungarian Machine Translation, *in* ‘Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC’2006)’, Genoa, Italy.
- Mikheev, A. (2002), ‘Periods, capitalized words, etc.’, *Computational Linguistics* **28**(3), 289–318.
- Miller, S., Crystal, M., Fox, H., Ramshaw, L., Schwartz, R., Stone, R., Weischedel, R. & Group, T. A. (1998), Algorithms that learn to extract information BBN: Description of the Sift system as used for MUC-7, *in* ‘Proceedings of MUC-7’.
- Mitchell, P. M., Santorini, B. & Marcinkiewicz, M. A. (1994), Building a large annotated corpus of English, *in* S. Armstrong, ed., ‘Using Large Corpora’, A Bradford Book, The MIT Press, pp. 273–290.

- Moore, R. C. (2002), Fast and accurate sentence alignment of bilingual corpora, in 'Proc 5th AMTA Conf: Machine Translation: From Research to Real Users', Springer, Langhorne, PA, pp. 135–244.
- Moscoso del Prado Martín, F., Kostic, A. & Baayen, R. H. (2004), 'Putting the bits together: An information theoretical perspective on morphological processing', *Cognition* **12/2004**; **94(1)**:1-18.
- Németh, L. (2002), Magyar Ispell – Válasz a Helyes-e?-re, in 'IV. GNU/Linux szakmai konferencia', Linux-felhasználók Magyarországi Egyesülete, pp. 99–107.
- Németh, L., Trón, V., Halácsy, P., Kornai, A., Rung, A. & Szakadát, I. (2004), Leveraging the open-source ispell codebase for minority language analysis, in 'Proceedings of SALT MIL 2004', European Language Resources Association.
- Och, F. J. & Ney, H. (2003), 'A systematic comparison of various statistical alignment models', *Comput. Linguist.* **29**(1), 19–51.
- Oravecz, Cs. & Dienes, P. (2002), Efficient stochastic part-of-speech tagging for Hungarian, in 'Proceedings of the Third International Conference on Language Resources and Evaluation (LREC2002)', pp. 710–717.
- Peterson, J. L. (1980), *Computer programs for spelling correction: an experiment in program design*, Vol. 96 of *Lecture Notes in Computer Science*, Springer.
- Pietra, S. D., Pietra, V. D. & Lafferty, J. (1997), 'Inducing features of random fields', *IEEE Tr. on Pattern Analysis and Machine Intelligence* **19**(4).
- Pléh, C., Németh, K., Fazekas, J. & Varga, D. (2011), Entropy measures and predictive recognition as mirrored in gating and lexical decision over multimorphemic Hungarian noun forms, in 'QMMMMD Workshop, University of California, San Diego (Jan. 15-16)'.

- Potthast, M., Barrón-Cedeño, A., Stein, B. & Rosso, P. (2011), 'Cross-language plagiarism detection', *Lang. Resour. Eval.* **45**(1), 45–62.
- Potthast, M., Stein, B. & Anderka, M. (2008), A Wikipedia-based multilingual retrieval model, in 'Proceedings of the IR research, 30th European conference on Advances in information retrieval', ECIR'08, Springer-Verlag, Berlin, Heidelberg, pp. 522–530.
- Preiss, J. & Stevenson, M. (2004), 'Introduction to the special issue on word sense disambiguation', *Computer Speech & Language* **18**(3), 201–207.
- Prekopcsák, Z. (2008), <http://www.kitchenbudapest.hu/projects/celebgraph>, Technical report, Kitchen Budapest Media Lab.
- Prószték, G. & Tihanyi, L. (1996), Humor – a Morphological System for Corpus Analysis, in 'Proceedings of the first TELRI Seminar in Tihany', Budapest, pp. 149–158.
- Prószték, G., Tihanyi, L. & Ugray, G. (2004), Moose: a robust high-performance parser and generator, in 'Proceedings of the 9th Workshop of the European Association for Machine Translation', La Valletta, Malta, p. 138–142.
- Rabiner, L. & Juang, B. H. (1986), 'An introduction to Hidden Markov Models', *IEEE ASSP Magazine* pp. 4–15.
- Rabiner, R. L. (1989), A tutorial on Hidden Markov Models and selected applications in speech recognition, in 'Proc. IEEE', Vol. 77, pp. 257–286.
- Racsmány, M., Conway, M., Keresztes, A. & Krajcsi, A. (2012), 'Inhibition and interference in the think/no-think task', *Memory and Cognition* **40**(2):168–76.
- Rácz, P. & Szeredi, D. (2009), Testing usage-based predictions on Hungarian vowel reduction, in '17th Manchester Phonology Meeting, Manchester, UK'.
- Ramshaw, L. A. & Marcus, M. P. (1995), Text chunking using transformation-based learning, in 'Proceedings of the Third ACL Workshop on Very Large Corpora'.

- Ratnaparkhi, A. (1996), A maximum entropy model for part-of-speech tagging, in ‘Proceedings of the Conference on Empirical Methods in Natural Language Processing’, University of Pennsylvania, pp. 133–142.
- Rebrus, P., Kornai, A. & Varga, D. (2012), ‘Egy általános célú morfológiai annotáció’, *Általános Nyelvészeti Tanulmányok* . to appear.
- Recski, G. (2010), NP-chunking in Hungarian, Master’s thesis, Eötvös Loránd University, Theoretical Linguistics.
- Recski, G., Rung, A., Zséder, A. & Kornai, A. (2010), NP Alignment in Bilingual Corpora, in N. C. C. Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner & D. Tapias, eds, ‘Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)’, European Language Resources Association (ELRA), Valletta, Malta.
- Recski, G. & Varga, D. (2009), ‘A Hungarian NP-chunker’, *The Odd Yearbook* .
- Recski, G. & Varga, D. (2012), ‘Magyar főnévi csoportok azonosítása’, *Általános Nyelvészeti Tanulmányok* . to appear.
- Recski, G., Varga, D., Zséder, A. & Kornai, A. (2009), Főnévi csoportok azonosítása magyar-angol párhuzamos korpuszban, in ‘VI. Magyar Számítógépes Nyelvészeti Konferencia’, Szegedi Tudományegyetem.
- Resnik, P. (1998), Parallel strands: A preliminary investigation into mining the web for bilingual text, in D. Farwell, L. Gerber & E. Hovy, eds, ‘Machine Translation and the Information Soup: Third Conference of the Association for Machine Translation in the Americas’, Springer, Langhorne, PA.
- Resnik, P. & Smith, N. (2003), ‘The web as a parallel corpus’, *Computational Linguistics* **29**(3), 349–380.

- Rosen, A. & Vavřín, M. (2012), Building a multilingual parallel corpus for human users, *in* N. Calzolari, K. Choukri, T. Declerck, M. U. Doğan, B. Maegaard, J. Mariani, J. Odijk & S. Piperidis, eds, ‘Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC’12)’, European Language Resources Association (ELRA), Istanbul, Turkey.
- Sekine, S., Grishman, R. & Shinou, H. (1998), A decision tree method for finding and classifying names in Japanese texts, *in* ‘Proceedings of the Sixth Workshop on Very Large Corpora’, Vol. 14(4):365-393.
- Shannon, C. E. (2001), ‘A mathematical theory of communication’, *SIGMOBILE Mob. Comput. Commun. Rev.* **5**(1), 3–55.
- Silfverberg, M. & Lindén, K. (2011), Combining statistical models for pos tagging using finite-state calculus, *in* ‘Proc. of the 18th Nordic Conference of Computational Linguistics NODALIDA 2011’.
- Simard, Michel & Plamondon, P. (1998), Bilingual sentence alignment: Balancing robustness and accuracy, *in* ‘Machine Translation’, Vol. Volume 13, no. 1, pp. 59–80.
- Solymosi, A. (2007), Tulajdonnév-felismerés, személynevek azonosítása magyar nyelvű szövegekben, Master’s thesis, Budapest University of Technology and Economics.
- Steinberger, R., Pouliquen, B., Widiger, A., Ignat, C., Erjavec, T., Tufiş, D. & Varga, D. (2006), The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages, *in* ‘Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC’2006)’, Genoa, Italy.
- Sutton, C. & McCallum, A. (2011), ‘An Introduction to Conditional Random Fields’, *Foundations and Trends in Machine Learning*. To appear.

- Szarvas, G., Farkas, R., Felföldi, L., Kocsor, A. & Csirik, J. (2006), A highly accurate Named Entity corpus for Hungarian, *in* 'Proceedings of International Conference on Language Resources and Evaluation'.
- Szarvas, G., Farkas, R. & Kocsor, A. (2006), A multilingual named entity recognition system using boosting and C4.5 decision tree learning algorithms, *in* 'Proceedings of the 9th international conference on Discovery Science', DS'06, Springer-Verlag, Berlin, Heidelberg, pp. 267–278.
- Szeredi, D. (2009), Functional phonological analysis of the Hungarian vowel system, Master's thesis, Eötvös Loránd University, Theoretical Linguistics.
- Talvensaari, T. (2008), *Comparable Corpora in Cross-language Information Retrieval*, Julkaisusarja A, University of Tampere, Department of Computer Sciences.
- Tiedemann, J. (2002), Uplug - a modular corpus tool for parallel corpora, *in* L. Borin, ed., 'Parallel Corpora, Parallel Worlds', Rodopi, Amsterdam, New York, pp. 181–197. Proceedings of the Symposium on Parallel Corpora, Department of Linguistics, Uppsala University, Sweden, 1999.
- Tiedemann, J. (2009), News from OPUS – A Collection of Multilingual Parallel Corpora with Tools and Interfaces, *in* N. Nicolov, G. Angelova & R. Mitkov, eds, 'Recent Advances in Natural Language Processing V', Vol. 309 of *Current Issues in Linguistic Theory*, John Benjamins, Amsterdam & Philadelphia, pp. 227–248.
- Tiedemann, J. & Nygaard, L. (2004), The opus corpus - parallel and free, *in* 'Proceedings of LREC'04', Vol. IV, Lisbon, pp. 1183–1186.
- Tjong Kim Sang, E. F. & Buchholz, S. (2000), Introduction to the conll-2000 shared task: chunking, *in* 'Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning - Volume 7', ConLL '00, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 127–132.

- Tjong Kim Sang, E. F. & De Meulder, F. (2003), Introduction to the CoNLL-2003 shared task: language-independent named entity recognition, *in* 'Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4', CONLL '03, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 142–147.
- Toral, A., Poch, M., Thurmair, G. & Pecina, P. (2012), Efficiency-based evaluation of aligners for industrial applications, *in* 'Proceedings of the 15th Annual Conference of the European Association for Machine Translation', European Association for Machine Translation, Trento, Italy.
- Tóth, K., Farkas, R. & Kocsor, A. (2008), 'Sentence alignment of hungarian-english parallel corpora using a hybrid algorithm', *Acta Cybern.* **18**(3), 463–478.
- Toutanova, K., Klein, D., Manning, C. & Singer, Y. (2003), Feature-rich part-of-speech tagging with a cyclic dependency network, *in* 'Proceedings of HLT-NAACL', pp. 252–259.
- Trón, V., Gyepesi, G., Halácsy, P., Kornai, A., Németh, L. & Varga, D. (2005), Hunmorph: open source word analysis, *in* 'Proceedings of the ACL 2005 Workshop on Software'.
- Trón, V., Halácsy, P., Rebrus, P., Rung, A., Vajda, P. & Simon, E. (2006), Morphdb.hu: Hungarian lexical database and morphological grammar, *in* 'Proceedings of LREC 2006', pp. 1670–1673.
- Tufts, D., Dienes, P., Oravecz, C. & Váradi, T. (2000), Principled hidden tagset design for tiered tagging of Hungarian, *in* 'Proceedings of the Second International Conference on Language Resources and Evaluation'.
- Turchi, M., Flaounas, I., Ali, O., Bie, T., Snowsill, T. & Cristianini, N. (2009), Found in Translation, *in* 'Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II', ECML PKDD '09, Springer-Verlag, Berlin, Heidelberg, pp. 746–749.



- Váradi, T. (2002), The Hungarian National Corpus, *in* ‘Proceedings of the Third International Conference on Language Resources and Evaluation’, Las Palmas, pp. 385–389.
- Váradi, T. (2003), Shallow parsing of hungarian business news, *in* ‘Proceedings of Workshop on Shallow Processing of Large Corpora, March 27 (SProLaC03)’, Lancaster, UK.
- Váradi, T. & Gábor, K. (2004), A magyar intex fejlesztéséről, *in* Z. Alexin & D. Csentes, eds, ‘II. Magyar Számítógépes Nyelvészeti Konferencia’, Szegedi Tudományegyetem, Szeged, pp. 3–10.
- Varga, D., Németh, L., Halácsy, P., Kornai, A., Trón, V. & Nagy, V. (2005), Parallel corpora for medium density languages, *in* ‘Proceedings of the Recent Advances in Natural Language Processing 2005 Conference’, Borovets. Bulgaria, pp. 590–596.
- Varga, D. & Simon, E. (2006), Magyar nyelvű tulajdonnév-felismerés maximum entrópia módszerrel, *in* Z. Alexin & D. Csentes, eds, ‘IV. Magyar Számítógépes Nyelvészeti Konferencia’, Szegedi Tudományegyetem, Szeged, pp. 32–38.
- Varga, D. & Simon, E. (2007), ‘Hungarian named entity recognition with a maximum entropy approach’, *Acta Cybern.* **18**(2), 293–301.
- Voutilainen, A. (1993), Nptool, a detector of english noun phrases, *in* ‘Proceedings of the Workshop on Very Large Corpora’, pp. 48–57.
- Waldenfels, R. v. (2011), Recent Developments in Parasol: Breadth for Depth and Xslt Based Web Concordancing with Cwb, *in* ‘Proceedings of Slovko 2011, Modra, Slovakia, 20–21 October 2011’, pp. 156–162.
- Yu, Q., Max, A. & Yvon, F. (2012), Aligning Bilingual Literary Works: a Pilot Study, *in* ‘Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature’, ACL, Montréal, Canada, pp. 36–44.

- Zhou, G. & Su, J. (2002), Named entity recognition using an HMM-based chunk tagger, *in* ‘Proceedings of the 40th Annual Meeting on Association for Computational Linguistics’, ACL ’02, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 473–480.
- Zhu, C., Byrd, R. H., Lu, P. & Nocedal, J. (1997), ‘Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization’, *ACM Trans. Math. Softw.* **23**(4), 550–560.
- Zséder, A., Recski, G., Varga, D. & Kornai, A. (2012), Rapid creation of large-scale corpora and frequency dictionaries, *in* ‘Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC’12)’, European Language Resources Association (ELRA), Istanbul, Turkey.
- Zsibrita, J., Nagy, I. & Farkas, R. (2009), Magyar nyelvi elemző modulok az UIMA keretrendszerhez, *in* ‘VI. Magyar Számítógépes Nyelvészeti Konferencia. Szeged’, pp. 394–395.

## Summary

The results presented in this thesis fall in three major categories: (1) Natural language processing (NLP) software components. (2) Language resources created by the software, typically (but not always) for Hungarian. (3) Any further research uses the software and the LRs have been put to.

The first three chapters cover no new material. Chapter 4 presents a class of systems solving the task of *morphological disambiguation* for Hungarian. Our results show that purely statistical systems can be effectively combined with rule-based morphological analysis, with a robust treatment of out of vocabulary words. We then present our morphologically disambiguated Hungarian webcorpus and frequency dictionary, and a list of applications in diverse fields. Chapter 5 presents a state-of-the-art Hungarian *named entity recognition* system employing maximum entropy modeling with a large feature set relying heavily on character n-gram based features. We highlight some applications of our system. Chapter 6 presents a state-of-the-art *noun phrase chunking* system for Hungarian. Chapter 7 presents our algorithms solving the task of *sentence alignment*. They are distinguished from alternatives mainly by performing, in the critical range of interest, an order of magnitude faster, without sacrificing accuracy. We present a long list of Hungarian and international corpus creation efforts and other applications of our tools, detailing those where the author was a participant. Chapter 8 documents a framework that integrates all our aforementioned NLP tools, and proceeds to present our *bixtext query* service based on this framework. The service is a web-based ‘crowdsourcing’ application with two interconnected goals: first, it is a tool for human translators for finding translations of phrases in context. Second, it is a vehicle for the semi-manual collection and fully manual validation of parallel corpus data.

## Összefoglaló

A disszertációban ismertetett eredmények három csoportba sorolhatók: (1) Nyelvtechnológiai szoftvereszközök. (2) Nyelvi erőforrások (korpuszok), amelyeket a fenti eszközök felhasználásával előállítottunk, magyar és más nyelvekre. (3) Kutatási eredmények, amelyek ezen eszközök és nyelvi erőforrások kiaknázására épülnek.

Az első három fejezet nem tartalmaz új eredményeket. A 4. Fejezetben olyan eszközöket mutatunk be, amelyek magyar nyelvre magas pontossággal oldják meg a *morfológiai egyértelműsítés* feladatát. Eredményeink megmutatják, hogy statisztikai alapú rendszerek eredményesen kombinálhatók szabályalapú morfológiai elemzéssel, a szótáron kívüli szavak robusztus kezelése mellett. Ezután bemutatjuk morfológiailag egyértelműsített magyar webkorpuszunkat és gyakorisági szótárunkat, és ezek alkalmazásait több tudományterületen. Az 5. Fejezetben bemutatjuk magyar nyelvű *tulajdonnév-felismerő* rendszerünket, amely maximum entrópia tanulást alkalmaz nagyméretű jegyhalmazokon. A 6. Fejezetben bemutatjuk state-of-the-art *főnévi csoport azonosító* rendszerünket. A 7. Fejezetben bemutatjuk *mondat- párhuzamosító* rendszerünket, amelyet elsősorban nagyobb feldolgozási sebessége emel ki a hasonló rendszerek sorából. Ezután nagyszámú magyar és nemzetközi korpuszépítési projektet és egyéb alkalmazást tekintünk át, amelyek rendszerünkre épültek, és részletesen is bemutatjuk azokat, amelyekben a szerző közreműködőként részt vett. A 8. Fejezetben elsőként többnyelvű adatok gépi feldolgozását végző keretrendszerünket mutatjuk be, amely integrálja a korábbi fejezetek eredményeit. Ezután bemutatjuk erre épülő *párhuzamos szöveg kereső* rendszerünket, amely egy web-alapú ‘crowdsourcing’ alkalmazás, két, egymással összefonódó céllal: egyrészt a fordítók munkáját megkönnyítő (fordítástámogató) eszköz, másrészt lehetőséget ad párhuzamos korpuszok félig automatizált építésére és teljesen manuális validálására.