Joint syntactic and morphological parsing of Crimean Tatar and Tuvan using the Kazakh treebank

Table of contents

Abstract

This work describes the development of a data-driven transition-based joint syntactic and morphological parser. We compare the performance of joint models over the traditional pipeline models and find an improvement in parsing quality. We show that joint parsing is applicable to cross-lingual language processing and parse Tuvan and Crimean Tatar corpora with the models trained on Kazakh data.

1. Introduction

Morphological and syntactic analysis are the stepping stones for more complex language processing applications, such as machine translation, information retrieval, question-answering, and many others. The quality of morphological and syntactic parsing directly influences the quality of end application output. Traditional language processing systems are organized as a consecutive pipeline: that is, the single best output of one component serves as input into the next component. The main limitation of this approach is that each part's performance depends heavily on the previous elements of the pipeline, and degrades quickly with accumulated errors. Consider a pipeline composed of a morphological tagger and a syntactic dependency parser: the morphological tagger must, with limited context, select the best analysis for each token. Should this selection be erroneous, the syntactic dependency parser is likely to produce wrong output as well. This problem is especially pronounced in richly inflected languages due to a large number of possible tagging options, and therefore, higher chances of making a mistake.

To address this issue, joint syntactic and morphological parsing was introduced. In this approach, the tagger and the parser are aware of each other's context, and use it to incrementally proceed with their decisions. Joint parsing has achieved the state-of-the art results and a significant advantage over the traditional techniques. This technique, however, is relatively new: it was first proposed in 2006 for Modern Hebrew and then expanded onto a few other morphologically rich languages (see section on joint syntactic and morphological parsing for more details). To the best of our knowledge, only one parser that uses joint models has been released by Bohnet and Nivre (2012).

This work aims at developing an open-source tool for joint morphological and syntactic parsing. The initial version will be released for further development and improvement by the community. We hope that this will promote the use of joint parsing and encourage further research. In addition to creating the software, we plan to test the tool on the languages used in previous research (to compare results), as well as the new languages (to compare against traditional solutions).

We also explore the applicability of the joint method to cross-lingual parsing. Cross-lingual techniques are applied to different tasks, such as sentiment analysis (Wan 2009), word sense disambiguation (Lefever and Hoste 2010), and others. The principal idea behind cross-lingual language processing is to apply the resources (e.g. corpora, treebanks, analyzers) of one language to process a different language, which is usually under-resourced. Although cross-lingual dependency parsing has been performed before, by e.g. Xiao and Guo (2014) and Tiedemann (2015), it did not benefit from joint morphosyntactic disambiguation. We experiment with a Kazakh joint model and apply it to parse two related languages, Crimean Tatar and Tuvan, seeking to answer the following questions:

1. Is it possible to build a Crimean Tatar and Tuvan dependency parser using data from Kazakh?
2. What performance can be achieved with a model that automatically selects the best morphological analysis for each word, i.e. is not joint?
3. Is it possible to train a morphological model for Crimean Tatar and Tuvan using Kazakh data?
4. What performance can be achieved with such a model, which is still not joint, but has more intelligent algorithms for disambiguation?
5. Is it possible to do full joint parsing of Crimean Tatar and Tuvan on Kazakh data? Would this approach improve results compared to the previous two approaches?

This work is organized as follows: in section 2 we give a literature review on joint syntactic and morphological parsing. In section 3 we talk about various aspects of natural language processing that are related to joint parsing: dependency parsing, machine learning in dependency parsing, and cross-lingual language processing. We then give information on the data we use and the languages with which we work in sections 4 and 5. Sections 6 and 7 describe in detail the two components of our joint system, the dependency parser and the morphological tagger. Section 8 revisits the workflow of the joint model. We report the experiment results in section 9, and draw the conclusion in section 10. The parser source code is hosted at GitHub[1].

2. Joint syntactic and morphological parsing

Joint syntactic and morphological parsing (also called joint disambiguation) has emerged as an effort to improve parsing accuracy for languages with rich morphology, for which the standard parsing techniques perform poorly. One of the first experiments discussing the benefits of joint processing was carried out by Reut Tsarfaty (2006). Tsarfaty explored the effects of joint morphological segmentation and part-of-speech tagging on parsing quality: the model that performed segmentation and tagging jointly had an advantage over the pipeline approach. The experiments were done with Modern Hebrew, where, as in other semitic languages, much of the useful information about syntactic relationships is encoded in the word's morphology. Conversely, the correct morphological analysis is easier to perform having some information about the syntactic role of the word.

Cohen and Smith (2007) make the next step in joint parsing and include syntactic relations into their model. They have trained two analyzers, the first of which includes segmentation and part-of-speech tagging modules, and the second performs constituency parsing. The analyzers are combined using the "product of experts" learning technique, which takes the product of independent probability functions to produce the final result. This work is also concerned with Modern Hebrew.

Goldberg and Tsarfaty (2008) have developed the first model that incorporated morphology and syntax as a single classifier, as opposed to the two separate classifiers of Cohen and Smith (2007). They perform morphological analysis is performed using lattice representation, where each path through the lattice represents one possible analysis of the sentence, and all paths are unweighted. The parser selects the best path using the context-free grammars learned from the corpus. Goldberg and Tsarfaty have experimented with different grammar complexity in regards to which grammatical features it includes. The paper reports improvements in parsing accuracy over the previous works of Tsarfaty and Cohen and Smith. Goldberg and Elhadad (2011) have

[1] github.com/Sereni/joint-parsing

used the PCFG-LA Berkeley parser to benefit from a stronger syntactic model in joint processing and improve on the previous results.

Further experiments with joint morphological and syntactic disambiguation have explored its effects on parsing both morphologically rich languages and languages with high ambiguity of word forms, such as Chinese and English. Li et al. (2011) have developed several joint parsing models for Chinese, which incorporate different features and use various pruning strategies to reduce search space. These models have shown improvement over the pipeline models for Chinese.

Lee and Smith (2011) have developed a joint model for morphologically rich languages, which outperforms the pipeline morphological and syntactic processors in Latin, Ancient Greek, Hungarian and Czech, using a limited set of features. A similar work has been done by Bohnet and Nivre (2012), who also proposed to use the joint technique for dependency parsing, as opposed to constituency parsing in the works previous to this and Lee and Smith's. Bohnet and Nivre develop a parser and experiment with Czech, German, English and Chinese, achieving state-of-the-art accuracy. Çetinoglu and Kuhn (2013) use this parser to process Turkish, and also report an improvement in parsing accuracy. Bohnet et al. (2013) further expand dependency parsing models by adding more sophisticated morphological information, and using word clusters to incorporate lexical information into the model.

Unsurprisingly, joint models have been proposed not only for modeling interaction between morphology and syntax, but also between syntax and semantics, and between semantics of single words and larger structures. Johansson and Nugues (2008) use separate syntactic and semantic models to provide a list of hypotheses, and use the best combination as a final parse. Dahlmeier et al. (2009) use a probabilistic joint model to classify sense of prepositions and semantic role of the corresponding prepositional phrase. Bordes et al. (2012) perform joint semantic parsing and word sense disambiguation, which is conceptually quite similar to syntactic parsing and morphological disambiguation. In addition, joint models may also deal with sub-word segments, and a number of works for Chinese word segmentation demonstrate improvement over the pipeline models (see e.g. Jiang et al. (2008), Kruengkrai et al. (2009), Sun (2011), and Zhang and Clark (2008)).

3. Methodology

In the following sections we give an overview of the various language processing methods central to our work.

3.1. Approaches to dependency parsing

This section outlines the main approaches to dependency parsing. Dependency parsing is the task of building a dependency structure for a given sentence, that is, determining syntactic heads and relations for each word.[2] The need for dependency parsing first arose in early machine translation applications, and one of the first modules has been implemented in the system of Centre d'Études pour la Traduction Automatique in 1961 (Hutchins 1979). Since then, dependency parsing has found many other applications in natural language processing and information retrieval, and has also evolved in terms of the techniques used. Modern approaches to dependency parsing can be classified into two large groups: grammar-based and data-driven.

Grammar-based parsers make use of dependency grammars to make parsing decisions. These grammars contain information on which syntactic structures are possible in a given language. Pure grammar-based approaches require that the grammar describes all possibilities beforehand, and a sentence that does not adhere to the pre-defined rules may not be parsed with such a grammar. The two main types of grammars are context-free and constraint dependency grammars.

Context-free grammars consist of a set of rules that map non-terminal and terminal symbols. A terminal symbol is such a symbol that can be the leaf of the dependency tree. In case of dependency grammars, which are a subset of context-free grammars as argued by Gaifman (1965), both terminal and non-terminal symbols are represented by words of the sentence, although they can be referenced by part-of-speech labels for shorter rule representation. Treating dependency grammar as a context-free grammars yields an important advantage: the traditional parsing algorithms, such as CYK (Younger 1967) may be used for dependency parsing.

Another type of grammar is called constraint dependency grammar, which describes linguistic constraints placed on the constituents. The constraints reduce the field of possible choices by stating which requirements must be satisfied on the part of

[2] Dependency parsing should not be confused with constituency parsing. The latter includes additional information about the phrasal categories, such as noun and verb phrases.

the head and the dependent for the dependency to be valid. To take an example from Kübler et al. (2009), any English noun should be a head to either a determiner or a genitive modifier. If this is not satisfied, then the noun must belong to the class of mass nouns. If none of these are satisfied, then the parse is not valid, and the parser should look for a different solution. If all the words in the sentence have been assigned a syntactic head without violating any constraints, the sentence has been successfully parsed.

Grammar-based methods were the first to be used for syntactic parsing. However, developing a practical grammar even for a subset of a natural language is a task that requires linguistic expertise and time. With the increase in computational power and the introduction of linguistic corpora, the data-driven approach has been proposed to address the shortcomings of grammar-based parsing.

The data-driven approach uses information from the corpora to make parsing decisions. The process consists of two parts: firstly, the parser needs to "learn" a language model from the corpus (we provide information on learning methods in section "Machine learning"); secondly, it must apply the model to new input and parse it. Depending on the internal structure of the parser, data-driven systems may be split into two categories: graph-based and transition-based parsers.

Graph-based parsers represent sentences as graphs. A well-formed parse must include all words of the sentence (which are vertices), and start at the root of the sentence. In practice, root is added to each sentence as an artificial token, so that all words have a syntactic head, including verbs that do not depend on any of the other words. The system first learns the possible trees from a treebank (syntactically annotated corpus). Then, in the process of parsing, many different trees may be constructed, each with its own probability score. The most probable analysis will be selected as final. Since finding the best dependency tree is equivalent to finding a maximum spanning tree of the graph (Kübler et al. 2009), standard graph processing algorithms can be used for parsing.

Another way to organize the internal logic of a data-driven parser is called transition-based parsing. Since this is the method used in our parsers, we will describe it in more detail in the next section.

3.2. Transition-based parsing

A transition-based parser is a system that works with configurations and transitions. It may be convenient to think of it as a finite-state automaton: the automaton states are similar to configurations, with the exception that the latter are structurally complex. The machine uses transitions to transfer from one configuration to another. The last configuration, if valid, represents one of the possible parses of the sentence.

In order to define what a configuration is, we should describe the internal structure of the parser. It features a stack and a queue (which is also called a buffer). The queue is a first-in first-out data structure that contains the words of the sentence. When a word is removed from the queue, it must be the leftmost word of the sentence. Similarly, should the word be put back into the queue, it is put before all others. The stack is a last-in first-out structure, which means that the latest addition to the stack will be removed first. Initially, the queue contains all the words of the sentence, and the stack is empty.

In addition to the stack and the queue, the parser has another container: a set of dependency arcs. Each arc has three components: a head, a dependent, and a dependency label, and therefore fully describes a dependency relation between two words. The set of arcs contains all the arcs that have been initialized by a given point of time. When parsing starts, this set is empty.

Returning to the configurations, one may think of them as a snapshot of the parser's state. It describes which words are currently on the stack and in the queue, and which arcs have been drawn. The parser uses the information about current configuration to determine which transition it should make next. A transition, in its turn, is an action that the parser may take to modify the current configuration. These actions are the following:

- SHIFT: move one word from the queue onto the stack;
- LEFT ARC: draw an arc from the first word in the queue to the first word on the stack, then remove the first word on the stack (because it already has a head);
- RIGHT ARC: draw an arc from the first word on the stack to the first word in the queue, remove the first word on the queue, and then move a word from the stack back into the queue.

Figure 1 provides a step-by-step illustration of the parsing process (example taken from Kübler et al. (2009). Note that the parsing begins with all words in the queue and an artificial root token on the stack. Parsing ends when the queue is empty. In order to decide which transition to make at each given step, the parser uses a model that it learns on a training corpus of annotated sentences. We give more information on machine learning and its methods in the next section.

Figure 1: Shift-Reduce algorithm example

Step	Stack	Queue	Relationship	Transition
1	root	Economic news had little effect on financial markets		
2	root, Economic	news had little effect…		shift
3	root	news had little effect…	Economic – attribute – news	left arc
4	root, news	had little effect…		shift
5	root	had little effect…	news – subject – had	left arc
6	root, had	little effect…		shift
7	root, had, little	effect on…		shift
8	root, had	effect on…	little – attribute – effect	left arc
9	root, had, effect	on financial markets.		shift
10	root, … on	financial markets.		shift
11	root, … financial	markets.		shift
12	root, … on	markets.	financial – attribute – markets	left arc
13	root, had, effect	on	on – prepositional complement – markets	right arc
14	root, had	effect	effect – attribute – on	right arc
15	root	had	had – object – effect	right arc
16		root	root – predicate – had	right arc
17	root			shift

3.3. Machine learning in transition-based parsing

As we have mentioned before, transition-based parsing is a data-driven approach, which means that the parser's algorithm partially depends on the data with which it works. Modifying the program behavior depending on input data is the domain of machine learning. Tom Mitchell defines learning as "improving with experience", meaning that the program yields better results as it processes more data (Mitchell 1997). In this section we will talk about the role of learning in transition-based dependency parsing.

A transition-based dependency parser operates by making transitions between its states, or configurations, to make way from the initial configuration to the final one. The dependency arcs drawn in the process constitute the parse of the sentence. At each step, however, the parser has more than one transition to choose from, and exploring all the possible combinations is not feasible. Selecting a correct transition given the current configuration can be approached as a classification problem. The goal of a classification task, in general, is to determine to which of the pre-defined categories a new sample belongs. In our case, the samples are parser configurations, and the categories are transitions extended with dependency labels: for example, the classifier would feature a separate category for each of the left-arc transitions with different relations, such as "LA-subject", "LA-verb", "LA-attribute", etc.

Before learning, each sample in the classification pool must be represented as a feature vector. The features used in transition-based parsing describe various parts of the configuration. Most often, they are compound and consist of an address and a attribute. The address points at one of the words in the configuration, for example, the first word on the stack, or the third word in the queue. The attribute describes one of the attributes of a given word, such as form, part-of-speech tag, etc. Therefore, some examples of the features would be "part of speech of the first word in the queue", or "dependency relation of the second word on the stack". Kübler et al. (2009) describe the features typically used in transition-based parsing, and we talk more about the features used in our parsers in sections on baseline and joint parsing experiments.

The parser can learn on data from a syntactically annotated corpus, or a treebank. However, to make use of the corpus data, the sentences must first be converted to a sequence of configurations and their corresponding transitions. This can be achieved with a simple algorithm (cited according to Kübler et al. 2009), provided that the corpus has information about each token's head and dependency relation. For each sentence in the corpus, first construct a dependency tree using the provided dependency information. Then, mimic the parsing process from an initial configuration to the final, with the exception that the parser always selects the correct transition for each configuration, based on information from the dependency tree. More specifically, it selects:

- LEFT ARC if there is an arc from the first word in the queue to the first word on the stack;
- RIGHT ARC if there is an arc from the first word on the stack to the first word in the queue, and all the arcs where the first word in the queue is the head have already been drawn;
- SHIFT if none of the above are satisfied.

Whenever the parsers selects an action, it is recorded as a category label for a given configuration. For arc transitions, the dependency relation label is also recorded. The pairs of configurations and their respective category labels constitute the parser's training data. Once the training data is available, the classifier may be trained with a variety of algorithms, such as logistic regression, support vector machines, k-nearest neighbors, and others (see e.g. Bauer and Kohavi (1999) for comparison). The obtained model is then applied to determine the next transition for each configuration in the process of parsing new input.

3.4. Cross-lingual approaches

Cross-lingual language processing is a family of different applications that build on a common idea: applying the resources created in one language to benefit from data in other languages. The similarities do not go much further, because each area of cross-lingual processing has its own notion about which resources it can use and for which purposes.

One very broad area of cross-lingual language processing is called cross-lingual, or multilingual information retrieval. The principal goal of CLIR is to establish links between the documents in different languages which talk about the same topic, and then process them together. The practical applications of CLIR include gathering information about one event covered by multilingual media, calculating document similarity (e.g. for cross-lingual plagiarism detection, see Ehsan and Shakery (2016)), or providing search results in multiple languages that the user may speak, given a single query in one of the languages. Cross-lingual lookup is often done through translating the query into other languages, later, the documents may or may not be translated back into the query language. Another approach is to operate with both the query and the corpus as abstract representations, which map words or concepts in different languages into vectors. This allows to search for similar entries using distance measures between the vectors.

Cross-lingual techniques are widely used in part-of-speech and morphological tagging, as well as syntactic parsing. The underlying motivation is the same: some languages, like English, have abundant corpora and other annotated resources, while others, especially minority languages, do not. Models or rule systems designed for one language may be successfully applied to others, allowing to bypass the costly process of creating and annotating corpora. The first cross-lingual parsing systems relied on parallel corpora, using the annotation projection method. This method is a two-step process:

- the sentence in the source language is annotated using the existing tools;
- the annotation is projected onto the target language using word-alignment information.

Wu (1997) proposed an algorithm for bilingual parsing of parallel corpora, which helps to extract linguistic constraints (i.e. “what is possible”) from two languages to be used in subsequent models. Yarowski et al. (2001) describe the first systems that perform cross-lingual tasks in the aforementioned sense, applying one language’s resources to the other. Their work concerns tagging, lemmatizing, and noun phrase bracketing (determining noun phrase borders). Hwa et al. (2005) address the problem of creating syntactically annotated corpora and propose an algorithm to create noisy treebanks based on the source language corpus and syntactic projections. They show that these treebanks can then be used to train statistical parsers, achieving performance comparable with commercial parsers. A more recent work by Tiedemann (2014) explores annotation projection on harmonized data sets (corpora with compatible structure). Tiedemann finds that his approach outperforms the more popular model transfer method, due to better prepared corpora used for training.

In its turn, the model transfer approach, which has been introduced after annotation projection, has gained popularity due to its softer requirements to input quality. The gist of this approach is to train a model on the source language corpus, and then use the same model to process texts in the target language. This process does not require parallel corpora, and the target language corpus can be of any quality, as far as it can be represented using the same features as the source corpus. In model transfer, the models often ignore the language-specific data, such as lemmas and word forms, and work instead with universal part-of-speech and morphological information. To compensate for the missing data, some workarounds may be used, for example, word clusters (Täckström et al. 2012) and vector representations (Xiao and Guo 2014).

Practically any task in natural language processing can be approached from cross-lingual perspective. Some of the fields we have not discussed are sentiment classification (Wan 2009), lexicon construction (Padó and Lapata 2005), spoken language understanding (Lefevre et al. 2010), anaphora resolution (Iida and Poesio 2011), and semantic role labeling (Van der Plas et al. 2011).

4. Data

This section describes the data we use for building and testing the parser. All our corpora use the ten-field CoNLL-U format as described by the Universal Dependencies project (Nivre et al. 2015). This format stores information about each token's form, lemma, part-of-speech tag, morphological information, syntactic head and syntactic dependency label. The information about ambiguous tokens is stored in their indices: the tokens with the same index are considered possible variations of a single token. In case of ambiguous token boundaries, we indicate the index span before the set of tokens. Consider a sentence in Crimean Tatar:

Men	*bu*	*adiseni*	*unutacaq*	*ekenim*.
I	this	accident.ACC	forget.FUT	COP.1SG

'I have to forget this accident'.

The token *unutacaq* can be interpreted in 4 different ways: as a finite verb, a verbal adjective, a verbal noun or a verbal noun with zero copula. The last analysis implies splitting into two different tokens. Figure 2 illustrates how this sentence is represented in our input format. Note that this example has not yet been parsed and does not belong to a gold corpus, therefore, it has no information about syntactic dependencies.

Our English parser has been trained on the Universal Dependencies English corpus (Silveira et al. 2014). This corpus consists of 16 662 sentences, which have been drawn from various web sources, such as news, reviews, questions and answers, e-mails and blogs. A set of 2077 sentences, or approximately 12%, has been reserved for testing. We have used alternative morphological tagging to conduct 1-best experiments: the testing set has been re-tagged using HunPos tagger (Halácsy et al. 2007). The tagger has been trained on the corpus' training set, where each word form has been assigned a complex tag. Each tag is a concatenation of the token's lemma, part-of-speech tag, and morphological analysis. Tagging accuracy compared to the gold corpus is 83.2%. N-best

Figure 2. Example of input file format

id	form	lemma	POS	POS2	morphology
1	Men	men	_	prn	pers|p1|sg|nom
2	bu	bu	_	prn	dem|nom
3	adiseni	adise	_	n	acc
4-5	unutacaq	_	_	_	_
4	unutacaq	unut	_	v	tv|ger_fut2|nom
4	unutacaq	unut	_	v	tv|fut2|p3|sg
4	unutacaq	unut	_	v	tv|gpr_fut2
4	unutacaq	unut	_	v	tv|ger_fut2|nom
5	unutacaq	e	_	cop	aor|p3|sg
6	ekenim	e	_	cop	aor|evid|p1|sg
7	.	.	_	sent	_

experiments have not been performed for English due to very high ambiguity, which yielded a number of sentence options unfeasible to analyze in our conditions.

Our Kazakh parser uses the treebank collected by Tyers and Washington (2015). It consists of 402 sentences from different domains: learners' books, folk tales, legal texts and Wikipedia articles. The alternative morphological analyses have been obtained via the morphological analyzer used in Apertium translation system plus a constraint-grammar filter of approximately 150 hand-written rules. The best analysis for 1-best has been selected with the disambiguation tool developed by Assylbekov et al. (2016).

For cross-lingual experiments, we use two small hand-annotated corpora of Tuvan and Crimean Tatar. They contain 115 and 150 grammar book sentences, respectively. The extra analyses have been obtained via the morphological analyzers which are part of Apertium Tuvan (Tyers et al. 2016) and Crimean Tatar systems. We made one small change to the corpora to match the tagset of the Kazakh corpus: all nsubj (nominal subject) and csubj (clausal subject) tags have been replaced with subj (subject), because the Kazakh corpus does not make these distinctions. There is no morphological disambiguator available for either of these languages, which means that we have not used automatically annotated disambiguated corpora.

Table 1 summarizes some of the parameters of our corpora.

Table 1. Parameters of the corpora

	tokens	**sentences**	**tokens per sentence**	**ambiguity (analyses per token, n-best)**
English	254830	16662	15.3	n/a
Kazakh	4516	402	11.2	1.28
Crimean Tatar (testing only)	855	150	5.7	1.77
Tuvan (testing only)	639	115	5.6	1.19

5. Languages

This section gives brief notes on the languages with which we work: Kazakh, Tuvan, and Crimean Tatar. All information, unless otherwise stated, is cited from Ethnologue (Lewis et al. 2016).

Kazakh is a Kipchak language, the national language of Kazakhstan. It is also spoken in Russia, China, Iran, Mongolia, Turkey, and Uzbekistan. The total number of speakers is about 12 million, for 98% of which Kazakh is the native language. Kazakh is a well-researched language with several corpora, numerous processing tools and translation solutions.

Tuvan is a Sayan Turkic language spoken primarily in Russia, and also in China and Mongolia by about 280 000 speakers. The language is used both at home and at work. There exist newspapers and television programs in Tuvan, and the Tuvan Wikipedia features 1 275 articles as of May 2016. Although a corpus for Tuvan has been planned (Voinov 2012), its website remains permanently unavailable at the moment of writing.

Crimean Tatar (not to be confused with Tatar) is a Kipchak language spoken in Krym, Turkey, Bulgaria, Romania, and Uzbekistan, by a total of over 500 000 speakers. The language is taught at some schools, but the majority of speakers are of older generation. There are some newspapers and TV programs in Crimean Tatar, and the corresponding Wikipedia features 4 700 articles. The Crimean Tatar corpus is currently available at http://korpus.juls.savba.sk/QIRIM/; it consists of news texts amounting to a total of 521 012 tokens.

6. Baseline experiments

In order to assess the capabilities of a joint parsing model, we have performed experiments with a baseline parser. This parser is purely syntactic, and the morphological information comes from a separate analyzer on the pre-processing stage. The internals of the baseline parser are similar to those of our joint parser: it is transition-based, and it learns statistical representations of the language from corpora in CONLL format. During the parsing process, the parser works with configurations and decides on the next transition to make at each step using the model it learned. For a detailed explanation of how transition-based parsing works, see section on transition-based parsing.

6.1. Experimental setup

We have experimented with several different classifiers and feature sets to obtain the best parsing results. Before we talk about these experiments in detail, let us consider the full feature set. The features used for training are determined with respect to each configuration, and include forms, lemmas, part-of-speech tags, morphological information, and dependency relationships of various tokens in the configuration. To learn more about how the feature vectors and their corresponding labels are extracted from the dependency treebank, see section on machine learning in dependency parsing.

Table 2 below summarizes the full set of our training features. STK and BUF signify the configuration's stack and buffer, respectively, and the indices denote the position of a given word on the stack or in the buffer, beginning with 0. LDEP and RDEP mean the leftmost and rightmost dependents of a given token. The morphological information is used in two forms: in the first, each piece (e.g. person, number, case) is stored separately; in the second, all pieces are stored as one string to account for their interdependency.

To read the table, use the row to determine the token's address in the configuration, and the column to get the attribute of this token. The cells containing a plus sign represent the features that have been used in the model, while empty cells represent other possible features in this space.

Table 2. Feature space for parsing

	form	lemma	part-of-speech tag	morhpological info (separate)	morhpological info (joint)	dependency relation
STK[0]	+	+	+	+	+	
STK[1]			+			
LDEP(STK[0])						+
RDEP(STK[0])						+
HEAD(STK[0])	+					
BUF[0]	+	+	+	+	+	
BUF[1]	+		+	+		
BUF[2]	+		+	+		
BUF[3]	+		+	+		
LDEP(BUF[0])						+
RDEP(BUF[0])						+

These features come from the list suggested by Kübler et al. (2009), which was extended to include extra positions and joint features. The joint features, which are not represented in the table above, are two simple features combined. These only concern the first items on the stack and in the buffer of each configuration, and take into account their part-of-speech tags and forms. Table 3 below illustrates our joint features. For example, we consider the part-of-speech tags of the first items on the stack and in the buffer, and store them together as an extra feature, in addition to considering them individually.

Table 3. Joint features

	BUF[0].form	BUF[0].pos
STK[0].form	+	+
STK[0].pos	+	+

In order to determine the best classifier and feature set, we have trained the parser using several learning configurations. Each configuration is a combination of a classifier and a feature set. Each classifier, in its turn, has a number of inner parameters, which have been tuned independently in each experiment to achieve the highest precision. The parameters were tuned using 5-fold cross-validation over the training set. The classifiers, as well as the internal tuning and cross-validation algorithms are implemented as part of the scikit-learn module for Python (Pedregosa et al. 2011). Table 4 lists the classifiers we used and their parameters.

Table 4. Classifiers for dependency parsing

Classifier	Parameters
Linear SVM with stochastic gradient descent	learning rate: constant average: True, False penalty: l1, l2, elasticnet alpha: 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6},
Decision tree	split quality criterion: Gini impurity, information gain splitting strategy: best, random class weight: balanced, equal
Nearest neighbors	weights: uniform, distance leaf size: 5, 10, 30, 50

We will now briefly describe each classifier.

Linear support vector machine (SVM) is a learning model that splits the data space into a number of regions, each region corresponding to one of the categories — in our case, transitions paired with their dependency labels. The model tries to divide the regions with the widest gap possible. The new samples are then “plotted” in the space, and the category assigned depends on where the sample is located. Linear SVMs are designed to work with two-class problems, therefore, they use an ensemble of classifiers to work with multiclass problems. Our particular classifier uses the one-versus-all strategy, where each class has its own classifier, which produces a confidence score for each new sample. The classifier with the highest confidence score assigns the label. The parameters we use for this classifier either define the behavior of the stochastic gradient descent algorithm ('learning rate', 'average') or tune the SVM ('penalty', 'alpha') to better adapt to our dataset.

Decision tree is a model that uses a set of "rules" to classify each sample. To provide an example, let us consider the famous Iris dataset (Fisher 1936), which describes three species of iris plants using some of their measurements. When learning, the model decides on a set of questions to ask about each new sample. In case of the iris flowers, the questions may be, "is the petal longer than 2 cm", or "is the sepal shorter than 1.5 cm". The answer to each question determines the path further down the tree. The leaf of the tree to which the model navigates through a series of questions contains the predicted class.

Nearest neighbors is a classifier that uses data points from the training set to predict the class of the new samples. Each new sample is placed in the data space represented by feature vectors. The model then selects a number of its closest neighbors (in our case, five) using the desired distance measure, and the new sample is assigned the class that the majority of its neighbors have.

6.2. Selecting the features

Although intuitively it may seem that more features should result in better model performance, this is not always the case. After having obtained surprisingly low scores on Kazakh data with the full set of features and a linear SVM — 47% LAS on the gold tagset as opposed to 76.8% reported by Tyers and Washington (2015) — we have decided to experiment with different feature sets. Our feature sets are all possible combinations of the following feature categories: form, lemma, part-of-speech tag, morphological information, and dependency relation. Note that we used all the positions listed in Table 2 for each of the categories. To provide an example, a reduced feature set that contains word forms, lemmas and morphological information uses the following features (note that the other columns are empty for this setup):

Table 5. Example of a reduced set of features

	form	lemma	part-of-speech tag	morhpological info (separate)	morhpological info (joint)	dependency relation
STK[0]	+	+		+	+	
STK[1]						
LDEP(STK[0])						
RDEP(STK[0])						
HEAD(STK[0])	+					
BUF[0]	+	+		+	+	
BUF[1]	+			+		
BUF[2]	+			+		
BUF[3]	+			+		
LDEP(BUF[0])						
RDEP(BUF[0])						

To find the best configuration, we have trained each of the three classifiers discussed above with each of the feature sets, and then calculated labeled attachment score on the testing corpus. The experiments were conducted on Kazakh data. Table 6 summarizes the results of these trials.

Table 6. Labeled attachment scores on the Kazakh test set using different feature sets and classifiers

feature set / classifier	SVM	Decision tree	Nearest neighbors
all features	54.3	23.3	56.9
form	18.7	34.4	25.0
POS	48.4	58.5	56.5
lemma	12.2	29.0	24.3
morphology	49.8	56.5	38.6
form + POS	53.9	60.2	58.7
form + lemma	33.4	45.8	32.5
form + morphology	59.0	65.8	41.5
POS + lemma	50.9	60.5	59.3

feature set / classifier	SVM	Decision tree	Nearest neighbors
POS + morhpology	68.0	74.0	60.0
lemma + morphology	54.3	62.3	40.4
deprel + form	17.4	11.6	20.9
deprel + lemma	17.8	2.9	30.9
deprel + POS	42.4	18.9	55.2
deprel + morphology	34.9	8.9	45.2
form + POS + lemma	53.2	58.6	59.3
form + POS + morph	69.6	73.9	59.0
form + lemma + morph	61.5	62.7	43.2
POS + lemma + morph	**71.6**	**74.4**	59.2
form + POS + lemma + morph	53.6	57.0	59.3
deprel + form + POS	48.5	16.2	54.6
deprel + form + lemma	34.7	12.9	34.1
deprel + form + morph	42.9	9.4	46.3
deprel + POS + lemma	49.2	13.7	56.7
deprel + POS + morph	59.8	21.9	58.8
deprel + lemma + morph	48.2	8.5	48.2
deprel + form + POS + lemma	52.7	21.8	56.9
deprel + form + POS + morph	65.6	18.5	**61.7**
deprel + form + lemma + morph	54.7	14.1	49.2
deprel + POS + lemma + morph	63.2	16.6	59.3

Here are a few observations about these data. All the classifiers exhibit similar dynamics when the feature set changes, with a notable exception: decision tree classifier delivers significantly lower quality results when the dependency relations features are present. Other classifiers are also negatively affected by these features, although not so drastically. One reason for such inefficiency of dependency relation features may be the limited availability of dependency data in the testing set. Indeed, the goal of parsing is to assign dependency relations, and therefore only the tokens that have already been assigned a syntactic head or a dependent have deprel features. Conversely, in the training set, each token has a dependency relation assigned, and the model may access extra features during training.

We would have expected the single-feature models (rows 2–5) to perform poorly, but surprisingly, information about part-of-speech tags or morphological features is sufficient to achieve reasonable parsing accuracy, up to 58.5 LAS with decision tree classifier and the POS feature set. The feature sets combining part-of-speech tags and morphological information perform the best, and are enhanced only slightly by adding information about either lemmas or word forms. Surprisingly, the model that features all four — part-of-speech tags, morphological information, lemmas and word forms — performs worse than either of the previous models. It is difficult to judge whether the combination of lemma and word form features reduce parsing quality, because the rest of the models either consist of three components (thus leaving a single-feature model for comparison), or contain deprel features, and both of these cases display sub-standard performance.

Unlike many learning models, the decision tree classifier can estimate each feature's contribution into the model. The composition of 100 most important features is shown in Figures 3 and 4.

The feature distribution is not surprising; indeed, as the feature set experiments suggest, morphology and part-of-speech features contribute the most to the model quality, while information about lemmas helps to refine the results. In fact, out of 8 lemmas that made it to the top 100 features, 4 are words, 3 — punctuation marks, and one lemma is empty, indicating that the requested position is not filled. As far as the locations are concerned, it is also natural that the stack top and the buffer front dominate the list of the most important features, because the transitions — shift, left arc, and right arc — concern these positions exclusively. However, even though other positions are never involved in the current transition, they supply some information about which transition should be made. The single most important feature of our model is "b0.pos=sent", which indicates that the front of the buffer is the end of the sentence.

Table 7 below contains the same experiment run on English data. Because of time constraints, we have trained the classifiers on a reduced set of approximately 2500

Fig. 3. Top 100 features for Kazakh decision tree (by location)

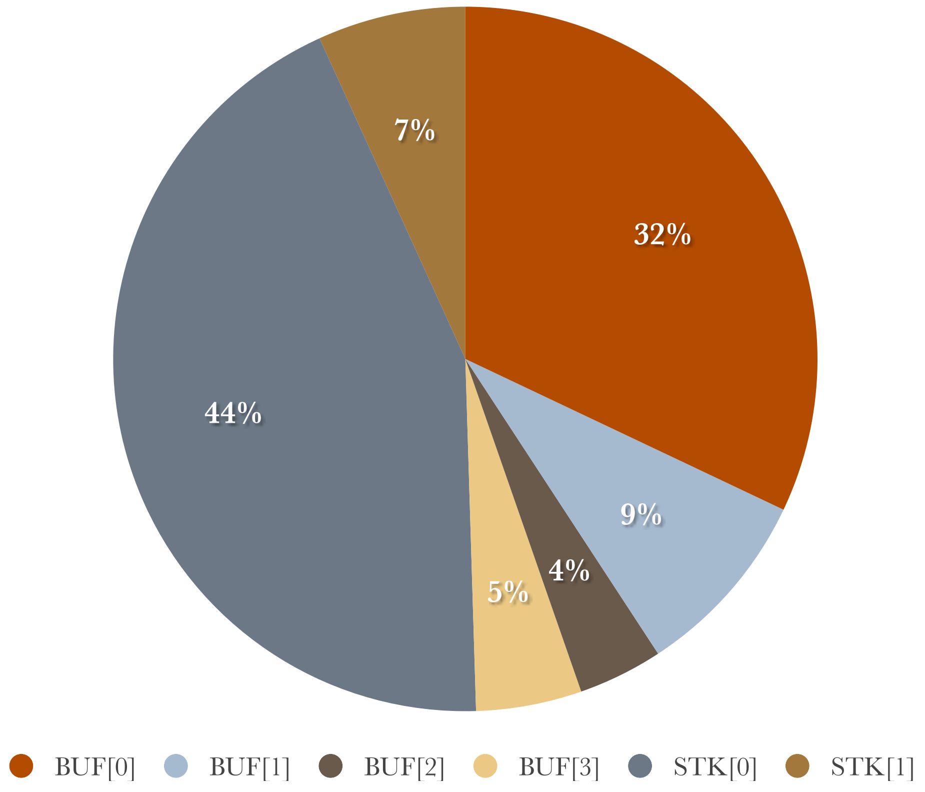
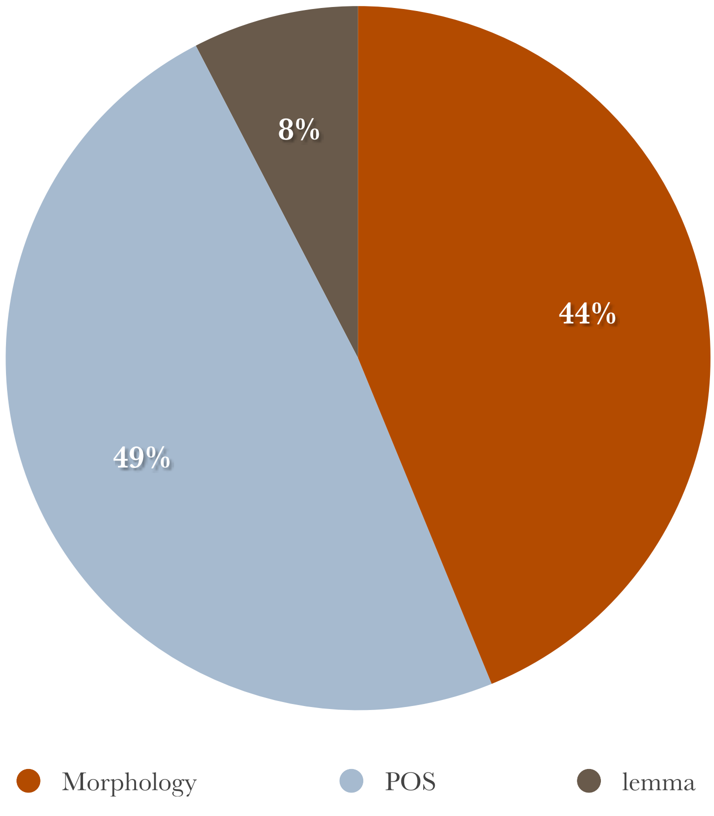


Fig. b. Top 100 features for Kazakh decision tree (by type)



sentences, or 1/5 of the full training set. In addition, because of the memory constraints, we have not experimented with the nearest neighbors classifier. In general, the classifier behavior is very similar to the Kazakh experiments, although dependency relation features seem to do even more harm in case of English. We have re-trained the best setup on the full training set and obtained the labeled attachment score of 67.0%.

Table 7. Labeled attachment scores on the English test set using different feature sets and classifiers

feature set / classifier	SVM	Decision tree
all features	60.1	18.2
form	43.6	42.2
POS	67.7	65.7
lemma	16.7	15.2
morphology	50.7	51.7
form + POS	73.0	70.4
form + lemma	49.8	44.3
form + morphology	63.6	61.2
POS + lemma	69.8	69.2
POS + morhpology	70.6	69.7
lemma + morphology	55.9	59.2
deprel + form	17.7	11.1
deprel + lemma	22.4	15.1
deprel + POS	50.5	19.9
deprel + morphology	33.8	8.8
form + POS + lemma	74.6	70.7
form + POS + morph	**75.5**	73.1
form + lemma + morph	65.7	63.8
POS + lemma + morph	72.8	**70.9**
form + POS + lemma + morph	74.2	**70.9**
deprel + form + POS	59.2	17.1
deprel + form + lemma	33.6	15.0

feature set / classifier	SVM	Decision tree
deprel + form + morph	47.2	16.2
deprel + POS + lemma	53.7	20.4
deprel + POS + morph	53.7	16.1
deprel + lemma + morph	38.7	16.0
deprel + form + POS + lemma	60.0	17.1
deprel + form + POS + morph	62.2	13.9
deprel + form + lemma + morph	47.7	17.0
deprel + POS + lemma + morph	55.6	18.1

6.3. *Results*

Using the best models described above, we have conducted three experiments to determine the boundaries of what the joint parser can achieve given our data. Each experiment has been run on the Kazakh and English corpora. In addition, we have performed two cross-lingual experiments: the Crimean Tatar and Tuvan test corpora have been parsed using the model trained on Kazakh data. For more details on corpora, see section on data. The experimental results are summarized in Table 8.

Table 8. Dependency parsing accuracy in baseline experiments, LAS

Language	Gold parses	1-best	N-best
Kazakh	71.7%	61.4%	61.8%
English	67.0%	54.9%	n/a
Crimean Tatar	79%	n/a	73.8%
Tuvan	71.9%	n/a	51%

In each experiment we use a different set of morphological tags. The first trial is run with gold standard morphological input from the treebank. The results represent an upper bound of what our parser could achieve if it knew the correct morphological analyses 100% of the time. The second trial uses the data from a morphological analyzer and disambiguator (Assylbekov et al. 2016 for Kazakh, HunPos for English), selecting one best hypothesis. The third trial is also run on the whole output of the morphological analyzer, using all possible analyses. Details on morphological analyzers can be found in the section on data. Each sentence is parsed and scored with labeled attachment score, which is the percentage of tokens that have been assigned both the correct head and the correct dependency relationship. LAS reported in the N-best column of the table considers the highest score the parser has achieved on any one of the available analyses.

7. The morphological model

Having performed the baseline experiments, we have set out to develop the joint syntactic and morphological parser. We have taken our syntactic parser as a base and added the capabilities for morphological disambiguation.

We treat morphological disambiguation as a classification task, similar to determining the best next transition in dependency parsing. In this case, the items to classify also are configurations, and the label assigned to each is a concatenation of the part-of-speech and the morphological tags of the first word in the buffer. In reality, the entity classified is the first word in the buffer, but because we use the features from other parts of the configuration, technically, we classify configurations.

We have chosen to perform disambiguation of the first word in the buffer. On the one hand, it is best to disambiguate as late as possible, so that the syntactic parser can benefit from additional information for as long as possible. On the other hand, all transitions in the syntactic parser assume that the tokens are already disambiguated, and the words that may participate in transitions are the first word in the buffer and the first word on the stack. Therefore, disambiguation happens right before the word can potentially participate in any transitions, but not earlier. We check if the buffer front needs disambiguation before predicting every following transition. We also accommodate for a special case when the analyzer splits the surface form into several structural tokens, which later form the dependency relations (for example, tokens 4-5 in [ref table format]). In this case, these tokens are "unwrapped", and both the buffer and the underlying sentence shift to make place for the extra tokens.

The features we use for morphological disambiguation are the same as for dependency parsing, with a minor change. Because we only disambiguate the token when it reaches the buffer front, the features concerning other items in the buffer were modified to work with ambiguous tokens in the following way:

• form: returns the form of the first analysis, or the unifying surface form for several deep tokens, if there are multiple;

• part-of-speech: returns the ambiguity class of the token, i.e. a concatenation of all distinct part-of-speech tags seen in the analyses for this token;

• morphological features: returns nothing if the token is ambiguous.

Similarly to the baseline experiments, we have performed the search for the best classifier and feature set pair. This time, we use 3-fold (instead of 5-fold) cross-validation to increase the number of samples in each category. Still, this remains an issue, because the classification is performed into 186 categories, and many of them have as few as 1-2 samples in the training set. The experiment was performed on Kazakh data. The results are presented in Table 9.

Table 9. Performance of the morphological disambiguator, label precision

feature set / classifier	SVM	Decision tree	Nearest neighbors
all features	47.7%	54.5%	41.5%
form	36.7%	52.1%	36.3%
POS	36.3%	45.9%	39.3%
lemma	26.0%	32.4%	27.6%
morphology	46.5%	54.5%	50.6%
form + POS	41.8%	53.3%	40.6%
form + lemma	33.1%	49.4%	36.0%
form + morphology	54.3%	68.0%	51.1%
POS + lemma	38.8%	49.6%	40.6%
POS + morhpology	66.0%	74.9%	54.9%
lemma + morphology	52.9%	60.7%	52.2%
deprel + form	35.4%	46.8%	30.6%
deprel + lemma	27.2%	31.5%	26.1%
deprel + POS	41.8%	46.8%	38.3%
deprel + morphology	54.0%	59.8%	48.3%
form + POS + lemma	42.0%	53.5%	41.2%

feature set / classifier	SVM	Decision tree	Nearest neighbors
form + POS + morph	67.8%	75.2%	54.6%
form + lemma + morph	60.8%	70.0%	52.3%
POS + lemma + morph	67.2%	74.6%	**56.2%**
form + POS + lemma + morph	68.8%	75.7%	55.5%
deprel + form + POS	45.7%	54.3%	39.4%
deprel + form + lemma	39.4%	46.4%	34.7%
deprel + form + morph	61.9%	69.5%	49.6%
deprel + POS + lemma	43.9%	50.9%	40.3%
deprel + POS + morph	68.4%	74.6%	53.3%
deprel + lemma + morph	57.7%	61.6%	51.1%
deprel + form + POS + lemma	47.1%	54.6%	41.5%
deprel + form + POS + morph	**69.8%**	**75.9%**	53.2%
deprel + form + lemma + morph	65.2%	70.0%	51.5%
deprel + POS + lemma + morph	69.0%	74.5%	53.9%

Although the training results for parsing and morphological tagging should not be directly compared, it is interesting to note that the dependency relation information does not impede the classifier performance in this case. Also, the nearest neighbors classifier has performed significantly worse than SVM and decision tree classifiers, perhaps, due to a small number of samples in each class. Another important detail to note is that the classifiers were not in any way restricted in the set of possible categories for each sample. In the real application, the class is selected from a set of tags assigned by the morphological analyzer, which may narrow down the possible choices and improve tagging quality.

As with the parsing experiments, we have re-trained the model with the best parameters (decision tree classifier, all features except lemmas), and obtained the morphological label precision of 59%. Part-of-speech tagging without the morphological analysis is accurate to 70.3%. When the same model is used for tagging Tuvan and Crimean Tatar, the classification scores are 40% and 30%, respectively.

Although a model of this quality is not suitable to perform stand-alone tagging, it achieves better precision when used in the joint model along with the restrictions from the morphological pre-processor, as we show in the section devoted to joint parsing results.

The decision tree classifier, once again, allows us to explore the most important features that make the biggest contribution to the model. Unsurprisingly, these features mainly deal with the properties of the first buffer item, which is the token under disambiguation (78 out of top 100), followed by the second buffer item and the second stack item (7 instances both), the first stack item (5), and the third buffer item (2). 91 of 100 features are either part of speech tags or morphological features of the token in question. This may seem counterintuitive, because these are precisely the features that we would like to predict. There are two things to consider: firstly, the buffer front does not always require disambiguation, because it may have been disambiguated in one of the previous steps, and then did not immediately move to the stack or become part of a dependency relation. Eliminating such cases from tagger training may be one way to improve its performance in the future. Secondly, if the buffer front is ambiguous, the part of speech features are automatically replaced with the "ambiguity class", which is a set of all possible part of speech labels that this token may receive. Sometimes, there would be only one possible part of speech, and the feature effectively becomes a simple part of speech feature.

Contrary to what we would expect, using language-specific features such as word forms and lemmas does not negatively affect the tagging quality in the cross-lingual setting. This is due to the fact that these features rank very low in classifier importance, except for a very few particular cases, most of which define punctuation and the root token.

8. Summary of the parsing workflow

The internals of our parser have been largely discussed in the sections on baseline experiments and morphological models. In this section we provide a summary of how the joint system works.

The parser accepts input corpus in CoNLL format. The required fields are id, word form, lemma, postag (language-specific part of speech tag), and feats (morphological information). If the postag field is empty, the parser will fall back to using cpostag (coarse part of speech tag), however, in this case the format should be consistent between the training corpus and the input data; otherwise, the performance will decrease due to the model working on an unknown set of tags. The input file format allows multiple analyses per token, which may be obtained with a naïve morphological analyzer.

In addition to the corpus to be analyzed, the parser requires a model, a vectorizer, and a feature set for both the parsing and tagging tasks, a total of six files. The models and vectorizers can be obtained via the training methods of the parser, provided a training and a development corpus. The vectorizer is an object that transforms categorical features (i.e. strings) into numerical features (0/1) via one-hot encoding. For example, if the hypothetical training samples are described by one feature — “color” — that may take values “red”, “blue”, and “green”, the vectorizer would transform them into three features “color=red”, “color=blue”, and “color=green”, which can take a value of 0 or 1.

The feature sets are Python lists serialized into JSON format, which contain names of features that should be used during parsing. All the features not on this list are filtered out during feature extraction. It is important to use the same feature set during parsing as when training the model (order of features is not important). The feature sets can be different for the parser and the tagger, because the feature vectors are used internally in each classifier to determine the next transition or the best tag, but the parser as a whole works on configurations, which are feature-agnostic.

The parser starts by loading the provided models. It then reads the sentences from the corpus, and passes each sentence into the parsing function. The dependency parser drives the process: it moves the state from one configuration to another by determining the next transition, until the buffer is empty. The morphological tagger works as a supplementary tool at each step, selecting the best analysis for the buffer front as described in the section above.

After the dependency parser has reached the end (i.e. buffer is empty), the last configuration is returned to the main function, which transforms it into CoNLL format for writing. The process is repeated until there are no more sentences in the corpus.

9. Joint experiment results

Having developed the joint parser, we have first trained two models on the Kazakh corpus. The model parameters we used were the best parameters found in the exhaustive search which was described in the previous sections. For the dependency parser, we used a decision tree classifier trained on the feature set that contains part of speech, lemma, and morphological features. The morphological tagger is also a decision tree classifier, and the feature set contains all features except lemmas: dependency relations, part of speech tags, morphological features, and word forms.

We have parsed the Kazakh, Tuvan, and Crimean Tatar corpora, and the results are presented in tables 10 and 11 below.

Table 10. Dependency parsing accuracy via a joint parser, LAS

	low estimate	wordform LAS	high estimate
Kazakh	58.5	63.2	67.6
Tuvan	40.9	58.4	62.0
Crimean Tatar	57.5	62.4	70.2

Table 11. Part-of-speech and morphological tagging precision, %

	Full tagging	POS only
Kazakh	59.0	70.3
Tuvan	68.2	84.1
Crimean Tatar	77.9	86.4

Here are a few observations about these data. First of all, the low and the high estimate in dependency parsing accuracy come from the automatic evaluation method that we use. In some cases, the morphological model decides on an analysis where the number of tokens differs from the gold analysis. This has happened 11 times (13%) while parsing Kazakh, 39 times (34%) with Tuvan, and 27 times (18%) with Crimean Tatar. We cannot automatically calculate labeled attachment scores in these cases, so we treat them in two different ways: for the low estimate, the LAS is considered to be 0 for each sentence that does not align with the gold sentence. For the high estimate, we ignore these sentences and calculate overall LAS without them.

We propose a workaround for the cases when the number of tokens in our analysis and the reference sentence do not coincide. Instead of comparing tokens line by line and using indices to point at dependency heads, we create a set of dependency relations between surface forms for both sentences, and then compare the sets. Each relation is a tuple of (surface form of head, dependency label, surface form of dependent). LAS for the sentence is then calculated as a number of dependencies common to both sets over the number of dependencies in reference. We still use the standard LAS calculations for sentences that allow it. Column “wordform LAS” in table 10 reflects the scores obtained with this metric.

To calculate morphological tagging and POS tagging precision, we have compared each token’s tag to its counterpart in the gold corpus. In case of token mismatch, i.e. when the number of tokens in the parsed sentence is different from the gold standard, the comparison proceeds as far as the end of the shorter sentence. The tokens after the one that caused a mismatch receive a score of zero, therefore, our scores may be slightly lower than true scores.

Table 12. Joint parsing scores compared to n-best scores, LAS

	n-best baseline	joint parsing
Kazakh	61.8%	63.2%
Tuvan	51%	58.4%
Crimean Tatar	73.8%	62.4%

Table 12 lists the baseline scores from n-best trials, and scores from joint experiments. Recall that in the n-best experiments we have constructed all possible sentences from a given combination of morphological analyses, and then calculated LAS as an average of the highest-scoring sentences. Comparing the scores against the baseline, we find a slight improvement in Kazakh parsing quality, a significant improvement in Tuvan, and a significant decrease in Crimean Tatar. Note that the joint score for Crimean Tatar is, in fact, comparable with other languages, but the parser has performed significantly better in n-best experiments for Crimean Tatar compared to Tuvan and Kazakh.

9.1. Error analysis

We have analyzed the errors made by the dependency parser and the morphological tagger. This section reports the common error patterns we discovered. We have observed, although to a lesser extent than in the pipeline models, the error accumulation effect: if the morphological tagger has selected an incorrect analysis for a given token, it will very likely enter an incorrect dependency relationship, which will at least partly affect the parse tree. The errors at this point may be incorrect tokenization (cases when one surface form is analyzed as several lemmas), incorrect part-of-speech label, or incorrect morphological analysis. Predictably, the parser also makes errors of its own, assigning incorrect head and/or dependency labels when the morphology has been determined correctly. We should note that the mistakes are not language-specific, and repeat across different corpora — which is not very surprising, provided we used the same model to parse them.

The first, and perhaps the most expected category of errors deals with part of speech ambiguity. Words like *bu / бо* 'this' and *o* 'that' can be classified as determiners, demonstrative pronouns, or personal pronouns (*o* as 'that' vs 'he'); *bir* 'one' can be a numeral or an indefinite determiner, the distinctions not always correctly made by our model. It also tends to select the substantive noun interpretation over an adjective, an error which has surfaced in the Tuvan and Kazakh corpora. Some of part of speech errors are common; others are made due to lack of training data. For example, the Kazakh word *көп* 'many' has once been misclassified as an adjective as opposed to a determiner, and once correctly classified as an adverb, but it has never occurred in the training corpus.

In cases when part of speech has been determined correctly, the morphological information may have not. The most common source of such errors is the distinction between verb forms. There are cases when passive transitive verbs have been classified as intransitive, and when participle tags have been assigned instead of (the correct) verbal adverb tags. These particular distinctions, however, have been up to debate in the annotation guidelines of the corpus, and also depend on the interpretation. Other morphological errors are more straightforward and reveal that the parser may be rather ignorant about the surrounding words. For example, the verb *басталды* 'started' has been classified twice as singular rather than plural, even though it has a correctly determined singular subject.

In general, having more context may improve parsing accuracy, although at a cost of considering more possibilities at each step. A common error that speaks in favor of this is finding multiple subjects in rather simple sentences — a pattern that is infrequent in training data, and that could have been better learned. Consider a sentence in Crimean Tatar, where three words — brother, every, and day — have been tagged as subjects (*er* was also erroneously tagged a noun):

Menim	*ağam*	*er*	*kün*	*şeerd*	*ola.*
I-POSS	older.brother	every	day	city-LOC	be-AOR

‘My older brother was in the city every day’

Our guess for such cases would be that the parser (especially having made an incorrect part of speech decision) assigns the relation that is most likely given a local context. It does not consider the likelihood of a 6-word sentence having 3 subjects, a knowledge which would significantly improve its performance.

The tendency to make localized decisions shows in another context: larger groups of conjuncts. In our gold corpora, the annotation standard states that one item in the group is tagged with its “real” dependency label, and the rest of the conjuncts receive a label *conj*, which shows their attribution to this group. If a conjunct is not directly near the tagged item — in case of three or more words excluding conjunctions — it receives a label that shows its relation to the head:

Пантомима,	*көлеңке*	*және*	*қуыршақ*	*театрлары*	*болған.*
mime	shadow	and	puppet	theater	be-PST

‘(There were) mime, shadow and puppet theater’.

In the gold corpus, both ‘mime’ and ‘shadow’ are marked as conjuncts, and ‘theater’ is a subject. Our parser has correctly determined that ‘shadow’ is a conjunct, but has labeled ‘mime’ as a subject in its own.

Finally, a small fraction of errors comes from rare categories, which have not been encountered often enough during training. For example, the only instance of a vocative relation in the Kazakh testing corpus has not been correctly determined, but it has only occurred twice in the training corpus. Another example is the dependency label ‘parataxis’, which signifies a relation between the main verb and the clause after a colon or a semicolon. This relation has no overt signs of coordination or subordination, and is therefore difficult to learn, especially on the 10 instances (0.3%) in the training corpus.

10. Conclusion

This work has been concerned with cross-lingual dependency parsing enhanced by joint morphological disambiguation. We have developed a joint syntactic and morphological parser, which is transition-based and operates with two independent classifiers. We have shown that it is possible to use the classifiers trained on Kazakh data to parse corpora in Crimean Tatar and Tuvan. After adding morphological disambiguation to the dependency parsing process, we have improved the parsing quality for Kazakh and Tuvan over the baseline scores. Our parser has been released for free modification and use; the current version is hosted at https://github.com/Sereni/joint-parsing/.

Bibliography

Assylbekov, Zh., Washington, J. N., Tyers, F. M., Nurkas, A., Sundetova, A., Karibayeva, A., Abduali, B., Amirova, D. (2016). A free/open-source hybrid morphological disambiguation tool for Kazakh. Presented at the 1st International Workshop on Turkic Computational Linguistics.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning, 36(1-2), 105–139.

Bordes, A., Glorot, X., Weston, J., & Bengio, Y. (2012). Joint learning of words and meaning representations for open-text semantic parsing. In International Conference on Artificial Intelligence and Statistics (pp. 127–135).

Cetinoglu, O., & Kuhn, J. (2013). Towards joint morphological analysis and dependency parsing of turkish. In Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013) (pp. 23–32).

Cohen, S. B., & Smith, N. A. (2007). Joint morphological and syntactic disambiguation.

Dahlmeier, D., Ng, H. T., & Schultz, T. (2009). Joint learning of preposition senses and semantic roles of prepositional phrases. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1 (pp. 450–458). Association for Computational Linguistics.

Debusmann, R. (2000). An introduction to dependency grammar. Hausarbeit Fur Das Hauptseminar Dependenzgrammatik SoSe, 99, 1–16.

Ehsan, N., & Shakery, A. (2016). Candidate document retrieval for cross-lingual plagiarism detection using two-level proximity information. Information Processing & Management.

Eisner, J. M. (1996). Three new probabilistic models for dependency parsing: An exploration. In Proceedings of the 16th conference on Computational linguistics-Volume 1 (pp. 340–345). Association for Computational Linguistics.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. Annals of Eugenics, 7(2), 179–188.

Gaifman, H. (1965). Dependency systems and phrase-structure systems. Information and Control, 8(3), 304–337.

Goldberg, Y., & Elhadad, M. (2011). Joint Hebrew segmentation and parsing using a PCFG-LA lattice parser. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2 (pp. 704–709). Association for Computational Linguistics.

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 (pp. 209–212). Association for Computational Linguistics.

Hutchins, J. (1979). Linguistic models in machine translation. UEA Papers in Linguistics, 9, 29–52.

Hwa, R., Resnik, P., Weinberg, A., Cabezas, C., & Kolak, O. (2005). Bootstrapping parsers via syntactic projection across parallel texts. Natural Language Engineering, 11(03), 311–325.

Iida, R., & Poesio, M. (2011). A cross-lingual ILP solution to zero anaphora resolution. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1 (pp. 804–813). Association for Computational Linguistics.

Jiang, W., Huang, L., Liu, Q., & Lü, Y. (2008). A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics. Citeseer.

Johansson, R., & Nugues, P. (2008). Dependency-based syntactic-semantic analysis with PropBank and NomBank. In Proceedings of the Twelfth Conference on Computational Natural Language Learning (pp. 183–187). Association for Computational Linguistics.

Kruengkrai, C., Uchimoto, K., Kazama, J. 'ichi, Wang, Y., Torisawa, K., & Isahara, H. (2009). An error-driven word-character hybrid model for joint Chinese word segmentation and POS tagging. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1 (pp. 513–521). Association for Computational Linguistics.

Lefever, E., & Hoste, V. (2010). Semeval-2010 task 3: Cross-lingual word sense disambiguation. In Proceedings of the 5th International Workshop on Semantic Evaluation (pp. 15–20). Association for Computational Linguistics.

Lefevre, F., Mairesse, F., & Young, S. J. (2010). Cross-lingual spoken language understanding from unaligned data using discriminative classification models and machine translation. In INTERSPEECH (pp. 78–81). Citeseer.

Lewis, M. Paul, Gary F. Simons, and Charles D. Fennig. (2016). Ethnologue: Languages of the World (19th ed.). Dallas, Texas: SIL International.

Li, Z., Zhang, M., Che, W., Liu, T., & Chen, W. (2014). Joint Optimization for Chinese POS Tagging and Dependency Parsing. Audio, Speech, and Language Processing, IEEE/ACM Transactions on, 22(1), 274–286.

Li, Z., Zhang, M., Che, W., Liu, T., Chen, W., & Li, H. (2011). Joint models for Chinese POS tagging and dependency parsing. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (pp. 1180–1191). Association for Computational Linguistics.

Manning, C. D., Raghavan, P., Schütze, H., & others. (2008). Introduction to information retrieval (Vol. 1). Cambridge university press Cambridge.

Mitchell, T. M., & others. (1997). Machine learning. McGraw-Hill New York.

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In Proceedings of the 8th International Workshop on Parsing Technologies (IWPT. Citeseer.

Padó, S., & Lapata, M. (2005). Cross-lingual bootstrapping of semantic lexicons: The case of framenet. In PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (Vol. 20, p. 1087). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, 12, 2825–2830.

Silveira, N., Dozat, T., Marneffe, M.-C. de, Bowman, S., Connor, M., Bauer, J., & Manning, C. D. (2014). A Gold Standard Dependency Corpus for English. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014).

Sun, W. (2011). A stacked sub-word model for joint Chinese word segmentation and part-of-speech tagging. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1 (pp. 1385–1394). Association for Computational Linguistics.

Täckström, O., McDonald, R., & Uszkoreit, J. (2012). Cross-lingual word clusters for direct transfer of linguistic structure. In Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies (pp. 477–487). Association for Computational Linguistics.

Tiedemann, J. (2014). Rediscovering Annotation Projection for Cross-Lingual Parser Induction. In COLING (pp. 1854–1864).

Tiedemann, J. (2015). Cross-Lingual Dependency Parsing with Universal Dependencies and Predicted PoS Labels. Depling 2015, 340.

Tyers, F. M., & Washington, J. (2015). Towards a free/open-source universal-dependency treebank for kazakh. In 3rd International Conference on Computer Processing in Turkic Languages (TURKLANG 2015).

Tyers, F. M., Washington, J. N., Bayyr-ool, A., Salchak, A. (2016). A finite-state morphological analyser for Tuvan. Presented at the LREC 2016.

Van der Plas, L., Merlo, P., & Henderson, J. (2011). Scaling up automatic cross-lingual semantic role annotation. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2 (pp. 299–304). Association for Computational Linguistics.

Voinov, V. (2012). Designing a tagset for annotating the Tuvan National Corpus. INTERNATIONAL JOURNAL OF LANGUAGE STUDIES (IJLS), 1.

Wan, X. (2009a). Co-training for cross-lingual sentiment classification. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1 (pp. 235–243). Association for Computational Linguistics.

Wan, X. (2009b). Co-training for cross-lingual sentiment classification. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1 (pp. 235–243). Association for Computational Linguistics.

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. Computational Linguistics, 23(3), 377–403.

Xiao, M., & Guo, Y. (2014a). Distributed Word Representation Learning for Cross-Lingual Dependency Parsing. In CoNLL (pp. 119–129).

Xiao, M., & Guo, Y. (2014b). Distributed Word Representation Learning for Cross-Lingual Dependency Parsing. In CoNLL (pp. 119–129).

Yamada, H., & Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In Proceedings of IWPT (Vol. 3, pp. 195–206).

Yarowsky, D., Ngai, G., & Wicentowski, R. (2001). Inducing multilingual text analysis tools via robust projection across aligned corpora. In Proceedings of the first international conference on Human language technology research (pp. 1–8). Association for Computational Linguistics.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n 3. Information and Control, 10(2), 189–208.

Zhang, Y., & Clark, S. (2008). Joint Word Segmentation and POS Tagging Using a Single Perceptron. In ACL (pp. 888–896).