UNIVERSITY OF BERGEN

MASTERS THESIS

Computational Support for Concept Blending applied to Musical Instruments

Author:
Andreas NAUSTDAL

Supervisor:
Prof. Bjørnar TESSEM

A thesis submitted in fulfilment of the requirements for the degree of Master

in the

Department of Information Science and Media Studies

December 1, 2017

University Of Bergen

Abstract

Faculty of Social Sciences
Department of Information Science and Media Studies

Master

Computational Support for Concept Blending applied to Musical Instruments

by Andreas NAUSTDAL

This thesis presents a concept blending implementation that suggests which properties of a known concept are most compatible to blend with another concept. The implementation uses Wikipedia descriptions of concepts as data source, and NLP tools such as WordNet and Stanford CoreNLP to create a representation of concepts and their properties. By joining the generalized properties of two concepts in a tree structure, we look for patterns in the data which correlate with properties that make sense to blend. A heuristic function based on these patterns is used to rank the subtrees to return top suggestions of features to blend between the two concepts.

Acknowledgements

I would like to thank Prof. Bjørnar Tessem for supervising and supporting me through the process. Thanks to Ankica Babic for helpful feedback and guidance in creating the project outline. I would also like to thank my family and friends for valuable feedback, and also room 634 for the good atmosphere.

Contents

List of Figures

List of Abbreviations

POS	**P**art **O**f **S**peech
WSD	**W**ord **S**ense **D**isambiguation
MRD	**M**achine-**R**eadable **D**ictionary
D	**D**epth of word in tree
BL	**B**ranch **L**ength from shared hypernym

Chapter 1

Introduction

1.1 Motivation/Background

Combining two concepts into a new concept that combine elements from each original concept is something humans can do rather easily. If we think of blending a horse and a bird, we can easily imagine a winged horse like the mythological Pegasus. Or when blending a car with a boat we can think of an amphibian boat. The earliest example of concept blending is the *Lion-man of the Hohlenstein-Stadel* which is around 32,000 years old (Turner, 2014). Conceptual blending is not just combining physical parts, but also abstract elements. For example, whenever we choose a day for an appointment next week, in our minds we manage to blend it in into the mental image of each day even though it happens in the future. We also blend the usual day-night cycle, the typical work-day and other appointments into these days. Then we can compare the days to see which one the appointment fits best into.

Although humans do conceptual blending unconsciously on a daily basis, formalizing the task of concept blending is not a straightforward process. The problem consists of three steps:

- Retrieving input data and creating a representation of the input space
- The creative strategy including the criteria for elements that will be blended and how the blending will be done
- Elaboration of the blend, including interpretation of properties and handling contradictions in the final blend.

For the concept blending algorithm to be useful, it is ideal that the blended parts make sense together in the new concept. Therefore, we want to find a way to decompose the concepts into properties containing enough information so that we can find patterns that identify suitable blending suggestions.

FIGURE 1.1: Pegasus is a blend of horse and bird. The wings have been merged onto the horse in a similar fashion to how birds have wings

The goal of this master thesis is to implement an algorithm that suggests elements that are suitable for transfer from one concept to another. Descriptions of the two concepts are retrieved from a public data source, Wikipedia. Natural language processing tools like WordNet and Stanford Parser are used to find which elements the concepts consists of. Although the concept blending algorithm may be used on any concept, we will focus mainly on the domain of musical instruments, with the goal of designing new instruments. Further the generated instruments will be evaluated on how useful they are in a musical context. The goal is to determine whether this approach can produce new instruments and discover how the algorithm can be useful to musicians and instrument makers.

1.2 Research Questions

Here we list three questions that we want to research through incremental improvements and evaluation of the algorithm.

- What are the characteristics of elements in a concept that are suitable for concept blending?

The reason we want to find these characteristics, is that we want to use them as a foundation for creating parameters to rank blending possibilities. On the other hand, characteristics of bad candidates can be useful for removing unwanted properties of concepts or penalizing suggestions deriving from these.

- How should we represent the concepts in order to find patterns of good elements for blending and detect contradictions indicating bad blends?

We want to find out which data are optimal for the input stage in order for the algorithm to work optimally and give the best suggestions. Can the concepts and their parts keep their original meaning and relations in the final results?

- Can a concept blending algorithm generate musical instruments that are reasonable and useful?

We want to evaluate the output of the algorithm to see if it is valuable in its environment.

The motivation for these questions is to uncover new knowledge about potential applications for the concept blending algorithm, how it can be used successfully and how it compares to other approaches.

Chapter 2

Literature review

In this chapter we will explain the fundamental terminology and present the academic literature we base our research on. Conceptual blending is a type of creative process, so we will start by explaining creativity.

2.1 Creativity

What is creativity? If we look at definitions of the word, we find explanations of general character. Cambridge Dictionary gives the following definition of creativity: *the ability to produce original and unusual ideas, or to make something new or imaginative.*

Creativity happens in different ways. Ideas may form spontaneously, or discovered through deliberate exploration of known domains. Sometimes a new invention is found because the right elements are available, and a combination of these create something new. For instance the first electric light was invented when Humphry Davy had access to the world's largest battery at the time, and passing current through a strip of platinum produced light. In other cases creativity is goal-driven. When electric light was commercialized it went through a creative process of choosing the right materials and finding an cost-efficient production method with the goal of making the light-bulb profitable.

Busse and Mansfield (1980) lists seven categories of creativity theories: psychoanalytic, gestalt, association, perceptual, humanistic, cognitive-developmental and composite theories. They also draw a line between convergent and divergent problems that the theories are applicable to. The difference there is convergent problems have one or few right answers while divergent has many possible solutions. One end is a more goal-driven and problem-solving type of creativity and the other end includes more open-ended activities such as art and conceiving novel ideas.

In the association theories, creativity is thought to come from associations. Arthur Koestler combines associationism and psychoanalytic concepts in what he called the *bisociation* theory (Busse and Mansfield, 1980). In bisociation, two independent matrices of ideas are combined, guided by subconscious processes. Koestler use the word *matrix* as a representation of abilities, habits, skills or any pattern of behaviour governed by fixed rules. (Koestler, 1964). Bisociation inspired the theory of conceptual blending by Gilles Fauconnier and Mark Turner (Fauconnier and Turner, 2002).

Boden (2004) lists three types of creativity: exploratory, transformational, and combinatorial. Exploratory is exploring the possibilities of a familiar domain. Transformational is changing rules and opening restrictions in order to break out of the known domain. Combinatorial creativity combines familiar concepts in a novel way, going outside known rules to create new ideas. An example of this can be found in Lego, where using a combination of bricks with different shapes and styles yield a lot more creative combinations than only using the standard 2x2 brick. (Popova, 2012) Conceptual blending has been used in efforts to create computational approaches to combinatorial creativity.

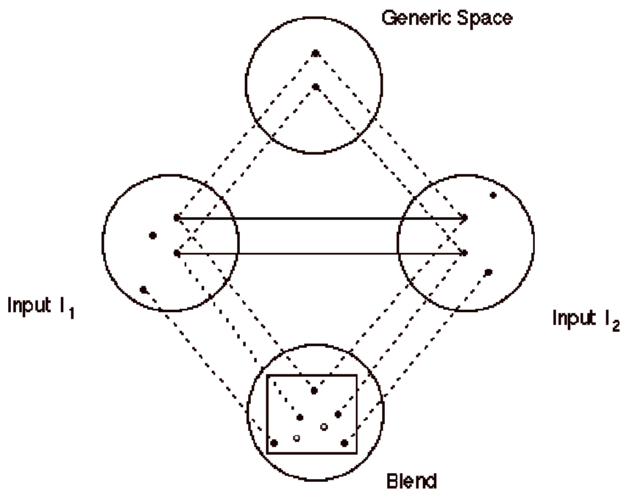

FIGURE 2.1: Conceptual blending model showing mental spaces (circles) and mappings of counterpart connections (solid lines) (Fauconnier and Turner, 1998b)

2.2 Conceptual blending

Conceptual blending is a theory proposed by Gilles Fauconnier and Mark Turner. They explain it as a cognitive operation where structure is projected from input mental spaces to a separate, "blended" mental space (Fauconnier and Turner, 1998a). The mental spaces contain structured elements and are interconnected to model dynamic mappings in thought and language. Conceptual blending can be explained using a model of four mental spaces consisting of two input spaces, the generic space and blend space (see Figure 2.1). Each input space is a partial model of the corresponding concept. The generic space contains what the input spaces have in common. Elements from the generic space are mapped onto the blend space, and may be blended with elements from the input spaces that are not mapped to the generic space.

2.3 Computational approaches

Efforts have been made to create a computational algorithm using the conceptual blending theory. Li et al. (2012) presents an algorithm with an emphasis on context-induced goals to prune the search space. Martinez et al. (2011) uses a logic-based approach not only in a mathematical setting but also in disparate problems such as rationality puzzles and noun-noun combinations. Besold and Plaza (2015) uses a similar logic-based approach but using amalgams developed by Ontañón and Plaza (2010).

2.3.1 Amalgams

One approach to conceptual blending is the use of amalgams (Ontañón and Plaza, 2010). An amalgam uses generalization of two concepts to make them compatible for blending through unification of the two generalizations. When parts of the concepts share a similar generalization, but are different, they are not unifiable. But they can unify with the *least general generalization*. To make the concepts unifiable, we can therefore generalize corresponding pairs of parts between the concepts. For instance, when creating an amalgam of a *red French vehicle* and a *German minivan*, we can make them unifiable by generalizing the first to a *red European vehicle*. When unifying we get a *red German minivan*. Or if we generalize the second to a *European minivan*, we can unify to a *red French minivan*.

2.4 Natural Language Processing

Natural language processing (NLP) is a field in computer science where computers interact with and process human natural language. We use NLP tools such as WordNet and Stanford coreNLP to process the concept descriptions into input spaces for blending. In the following section we will shortly describe important NLP concepts used throughout this thesis.

2.4.1 Machine-readable dictionary

A machine-readable dictionary (MRD) is a lexical database that can be queried by a computer program. The quality of the representation of concepts depend on the number of words in the database. The more words there are in the domain of musical instruments, the more information we can get from the Wikipedia descriptions we aim to blend. We use the WordNet database, as it was the only MRD we found that was large enough for our application. (Fellbaum, 1998)

2.4.2 Synsets

Words that have the same meaning are called synonyms. A synset is a set of synonyms. An ambiguous word can have many different synsets. A *crane* is not only a bird, it is a device that lifts and moves objects. The most extreme case is the word *break* which can have 75 different meanings.

2.4.3 Hypernyms

Words belong in categories. A rook belongs to the category chess piece. Chess piece belongs to the category of pieces of any type of board game. If we continue generalizing we get the categories game equipment, equipment, instrumentality, artifact, unit, physical object and physical entity until we finally reach entity. The category something belongs to is called a *hypernym*, and a specific instance of this is called a *hyponym*. A hypernym is the hypernym of a hyponym. Throughout the thesis we will use the terms hypernyms and categories interchangeably.

2.4.4 Part-of-speech

Words can be categorized based on grammatical properties. These categories are called part-of-speech and include nouns, verbs, adjectives, adverbs, pronouns, prepositions, conjunctions, interjections, numerals, articles and determiners.

2.4.5 Word sense disambiguation

A word can have different meanings called *word senses*. *Word sense disambiguation* (WSD) is the problem of identifying the correct word sense when a word has different meanings in a sentence or text. Approaches are commonly divided into supervised, unsupervised and knowledge-based WSD (Navigli, 2009).

Supervised methods train a classifier using a set of texts where the words have been labeled with the correct sense. (Navigli, 2009).

Unsupervised methods are based on data sets of unlabelled texts. (Navigli, 2009).

2.4.6 The Lesk Algorithm

The *Lesk algorithm* is a knowledge-based method where a machine readable dictionary is used to look up the senses and definitions of each word. The definitions which have the most overlapping words with definitions of nearby words are chosen to be the correct definition. (Lesk, 1986) A variant of this method is called *Simple Lesk* (Kilgarriff and Rosenzweig, 2000). It differs from the original by returning the definition which has the most overlapping words with the surrounding sentences. Since we are using Wikipedia articles, an approach for extending this technique could be to compare versions of the article in different languages, and look for overlapping senses in these versions by using multilingual versions of WordNet. For each word in English, look for the sense that has the highest frequency in the versions of the other languages.

2.4.7 Stop words

Stop words is a concept where unwanted, regular appearing words are removed from the sample text. Words such as *I*, *is*, *do* or other common binding words are discarded and replaced by empty strings. By removing these stop words the only words remaining are words that are more likely to be significant.

Chapter 3

Research methods and methodologies

In this chapter we describe our methods for development and research.

3.1 Personal Kanban

In order to keep my work organized and productive, I used the method called Personal Kanban. To visualize the workload, Personal Kanban uses a value stream which are columns that divide the work based on the progress of each task. I used the simplest one that divides the work into three columns. The first is the project backlog which includes all the tasks that awaits production. The middle column is the tasks currently being worked on, which should be a limited number of tasks so that they are not getting stuck half-finished and distracting the task one is actually doing. The last column is the tasks that have been completely done so that there are no reason to be distracted by it anymore. (Benson, 2009)

To keep a workflow where I can spend my energy on making sure the model, view and controller components is interacting well together, I limited my task number to 3. This way I could select one task from each component that depended on each other, while unfinished tasks would not be slowing down the workflow.

To review my progress, I had a daily evaluation inspired by the stand-up meeting used in XP (Wells, 1999) where I answered the following questions:

1. What was accomplished yesterday?
2. What will be attempted today?
3. What problems are causing delays?

3.2 Research methods

Simon (1996) separate design science from natural science. March and Smith (1995a) state that research activities in design science and natural science has different intentions. Design science intends to build and evaluate artifacts, while natural science intends to theorize and justify theories.

3.2.1 Design Science

Research Activities

		Build	Evaluate	Theorize	Justify
Research Outputs	Constructs				
	Model				
	Method	X	X		
	Instantiation				

FIGURE 3.1: Our research activities and outputs, following the research framework of March and Smith (1995b)

In our research we follow the design science activities of the research framework of March and Smith (1995a) (see Figure 3.1). March and Smith (1995a) divides the types of research artifacts into constructs, model, method and instantiation:

- *Constructs* are domain-specific vocabulary. For example graph theory consist of constructs such as nodes and edges.
- *Models* describe the relationships between constructs. For example a directed acyclic graph, which has the constructs nodes and directed edges, describe the relationship involving edges only going from nodes to other nodes in the forward direction.
- *Methods* consist of steps which are used to perform a task. Methods can be based on constructs and models or used to translate from one model to another.

- An *instantiation* is an artifact that can be used in an environment. They use constructs, models and methods to form a system that demonstrates the effectiveness of their artifacts.

In our research we have built and evaluated a method for concept blending of musical instruments and other domains. Our method is based on the amalgam framework by Ontañón and Plaza (2010). It contains a model describing a generalization space. We apply this model with the constructs *synsets* and *hypernyms*, which we are able to generalize through WordNet.

Chapter 4

Development

In this thesis we have developed an application which provide suggestions of which properties that are most compatible for blending between two concepts of a domain such as musical instruments. We present an overview of the tools and technologies, data representations and algorithms used in the application and describe how the components interact with each other. A simplifed overview of the final application, its components and data flow can be seen in Figure **??**. Deeper discussion and analysis will be presented in section 5.

4.1 Tools and technologies

Our application was developed in JavaScript incorporating a set of tools and technologies listed in the following section.

4.1.1 Wikipedia

Wikipedia is a large online encyclopedia, consisting of over 5 million english articles (Wikipedia, 2017). We use it as a data source for descriptions of the concepts. The application queries Wikipedia for the concept or instrument

TABLE 4.1: An overview of key components of the application

Algorithms	**External tools**	**Data representations**
Process Wikipedia Abstract	WordNet	Wikipedia abstract
Identify noun phrases	Stanford parser	Concept
Word sense disambiguation	Stanford POS tagger	Properties
Get generic space	Porter stemming algorithm	Generic space
Rank blending suggestions	Stop word list	Subtrees

we want to blend. The first section in the article, which we refer to as the abstract, is processed to retrieve properties that represent the concept.

4.1.2 WordNet

WordNet is a machine-readable dictionary containing 155,287 unique English words and 117,659 synsets. (Fellbaum, 1998) (*WordNet Statistics*)
The application use WordNet in to ways when retrieving language data:

- Retrieving synsets by querying the WordNet database with words to get the synsets of each word.
- Retrieving hypernyms by querying the WordNet database with synsets to get all hypernym ancestors of that synset.

4.1.3 Stanford coreNLP toolkit

The Stanford coreNLP toolkit is widely used for natural language processing. It includes tools for tokenization, sentence splitting, lexicalized parsing, part-of-speech tagging, morphological analysis, named entity recognition, syntactic tagging, coreference resolution and more. (Manning et al., 2014) We use two of their tools, a lexicalized parser and a part-of-speech tagger.

4.1.4 Parser

A lexicalized parser takes a sentence and divides it into grammatical units such as clauses, phrases and words. A treebank is a set of texts annotated with a syntactic or semantic sentence structure, which is used as training data for the parser. The English Stanford parser is trained using the Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993).

Our application uses the Stanford parser to find noun phrases in the concept descriptions with the goal of using them as properties. A noun phrase has the same grammatical function as a noun. The last word is a noun and the first words modify it. An example of a noun phrase is *five-string electric bass guitar*, where the words *five-string electric bass* are modifying the meaning of the noun *guitar*. The WordNet database contain noun phrases also and may return synsets which carry the meaning of the whole phrase.

4.1.5 POS tagger

A POS tagger takes a text and tag each word with its part-of-speech. We use it to tag the words in the concept descriptions. Each synset retrieved in WordNet comes with a POS tag, but words may have multiple synsets of different POS. Since we have tagged the word, we can exclude synsets of the wrong POS.

4.1.6 Porter stemming algorithm

The Porter stemming algorithm was written by Martin Porter and is a tool for removing morphological and inflexional endings of words (Porter, 1980). For example, the word *fishing* returns the stem *fish*. The application use the stemming algorithm on each word of the Wikipedia abstract except stop words to get possible stems. WordNet does not recognize all forms of the word, so by stemming we can get access to synsets of the stem.

4.1.7 JavaScript Frameworks and Libraries

The implementation was written in JavaScript using the frameworks AngularJS and Node.JS. The algorithm uses the following libraries from the Node Package Manager (NPM):

wordnet-magic

wordnet-magic is a Node.JS implementation of Princeton's WordNet lexical database for the English language (Burckhardt, 2016). We use version 3.1 of the database.

stanford-simple-nlp

stanford-simple-nlp is a Node.JS wrapper for version 3.3.1 of StanfordCoreNLP (Kim, 2014).

stemmer

stemmer is a Node.JS implementation of the Porter stemming algorithm (Wormer, 2017).

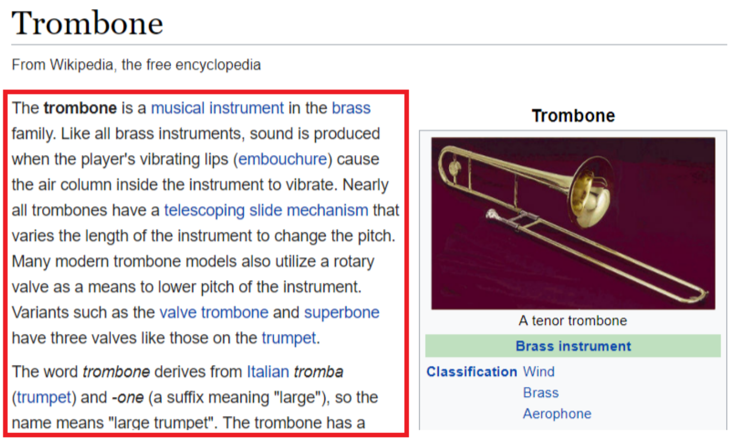

FIGURE 4.1: The Wikipedia abstract of the instrument *trombone*.

4.2 Data representations

4.2.1 Wikipedia Abstract

The Wikipedia abstract contains a text describing the concept. See 4.1 for an example of an abstract.

4.2.2 Concept

We have two concepts we want to blend. Each concept store the following data:

- A name which we will use to find the corresponding Wikipedia article
- A Wikipedia abstract
- The Wikipedia abstract without stop words
- A list of properties.

4.2.3 Properties

Each property store the following data:

- The original word
- A list containing the original word and possible stems found by the stemming algorithm.
- The part-of-speech such as noun, verb or adjective which have been tagged by the POS tagger

- A list of synsets, any unique meaning that the word or the stem of the word may have. Can be pruned to store only synsets of the same part-of-speech that the POS tagger found.
- A reference to the synset of the most likely definition found by the *get most likely definition* method
- A list of hypernyms, from the most specific to the most general, starting with the direct ancestor of the synset.

4.2.4 Generic Space

The generic space contains a list of synsets and hypernym synsets that exist in both concepts. For example if the first instrument has keys, and the other has pedals, the generic space will contain their shared hypernym *lever*.

4.2.5 Subtrees

Each concept has a list of subtrees containing unique properties and their hypernyms that are subsuming but are not themselves in the generic space. They are given a compatibility score by the *rank compatible blending suggestions* method. The subtree is assigned a depth, a branch length and a penalty value. The depth value is assigned the depth of its least general hypernym in the generic space. The branch length is the number of nodes in the subtree. The penalty value is equal to a penalty constant if the property is or has a hypernym in a penalized category.

4.3 Algorithms

4.3.1 Process Wikipedia abstract

This method remove unwanted artifacts from the Wikipedia abstract, preparing each word so that WordNet is able to understand them.

4.3.2 Identify noun phrases

This method identify possible noun phrases in the abstract. It also recursively search the noun phrase for smaller noun phrases by removing the first word.

4.3.3 Word sense disambiguation

The application use the *word sense disambiguation* method to find the synset that most likely carry the correct definition of the word. The method takes the following input:

- The ambiguous word in question
- The different definitions of the word
- A list of scoring words, which is made up of the abstract words and a set of words relating to the context (musical instruments)

4.3.4 Get generic space

This method takes returns the intersection of the properties and hypernyms of the two concepts.

4.3.5 Rank compatible blending suggestions

Each subtree are given a compatibility score. This is calculated by the depth minus branch length minus penalty. They are then sorted in a list by this compatibility score so that the best blending suggestions rise to the top.

Chapter 5

Concept Blending Algorithm

Our implementation of a concept blending algorithm consists of three steps. The first is retrieving the input data consisting of concept descriptions from a data source such as Wikipedia. The second step is creating the generic space, the structure deriving from both inputs. This is done by retrieving generalizations of the nouns from each concept, creating a tree structure of shared hypernyms. The third step is finding suggestions of elements to blend by using a heuristic function to rank the elements using patterns based on characteristics we have found in good blending examples.

5.1 Preparing input data

5.1.1 Finding useful properties

In our approach to concept blending, the elements that make the input spaces are nouns from the Wikipedia description. The reason for this is that we can generalize the nouns by retrieving their hypernyms using WordNet. We focus on nouns because they are more likely to be physical elements than verbs, adjectives or any other part-of-speech. Verbs are actions, occurrences or state of beings, rather than elements. Although verbs can be generalized with hypernyms as well, we can not compare the generality of verbs and nouns, since verbs usually have fewer hypernyms than nouns. The characteristics we look for are based on the distances in the hypernym tree. The distances we look for are the number of hypernyms between the word, the least general shared hypernym and the most general shared hypernym. If the compatibility score of the blending suggestions is based on how specialized the word is, it will favour nouns over verbs, since in WordNet verbs usually have fewer hypernyms than nouns, and it would not make sense to compare verbs and nouns this way.

An optimal solution may retrieve the properties and all the relations between the words to create a more complete representation of the concept, and not just elements. We focused on properties in the form of nouns since we wanted to validate the use of WordNet to create amalgams of musical instruments. In further research on a more complete algorithm for conceptual blending, it may be possible to extend the representation, modelling relations between the elements in order to maintain its original context in the blend space. It could also be possible to generalize the relations using WordNet, but research should be done on how these relations should be extracted, scored and blended.

FIGURE 5.1: A piano

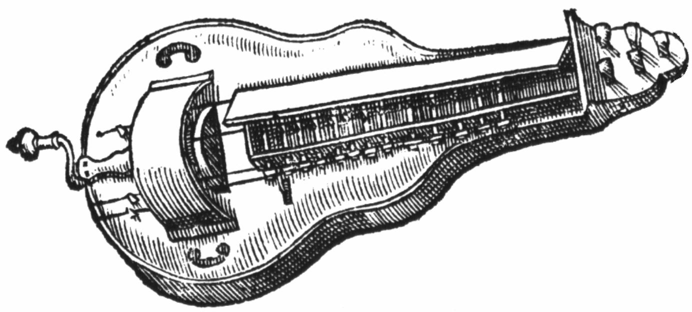

FIGURE 5.2: A hurdy gurdy. Like other acoustic instruments it has a hollow cavity that enables us to hear the sound coming from the vibration of the strings

5.1.2 Extracting property candidates from Wikipedia

Lets say we have two concepts that we want to blend, for example *piano* and *hurdy-gurdy* (seen in Figure 5.2). We can retrieve the abstracts using a HTTP call to the wikipedia API:

```
http://en.wikipedia.org/w/api.php?action=parse&page=
conceptName&prop=text&section=0&format=json&redirects=1&
callback=?
```

The parameter *conceptName* stands for the name of our concept. By extracting the first paragraph element (<p>) in the returned JSON string, we can retrieve the raw abstract. For hurdy gurdy this abstract is:

> The hurdy-gurdy is a stringed instrument that produces sound by a hand crank-turned, rosined wheel rubbing against the strings. The wheel functions much like a violin bow, and single notes played on the instrument sound similar to those of a violin. Melodies are played on a keyboard that presses tangents, against one or more of the strings to change their pitch. Like most other acoustic stringed instruments, it has a sound board and hollow cavity to make the vibration of the strings audible. Most hurdy-gurdies have multiple drone strings, which give a constant pitch accompaniment to the melody, resulting in a sound similar to that of bagpipes.

In order for the algorithm to be able to provide suggestions for the best properties to transfer between two concepts, we need to find properties in abstract that may represent the concept. To be able to query WordNet with words from the text we process the abstract with the steps seen in Procedure 1. This involves removing unwanted symbols and artifacts. Initially the descriptions consists of a lot of common words we are not interested in called stop words. We use a list of stop words to identify and remove them. But before they are removed, we send the original abstract to the Stanford part-of-speech (POS) tagger to have each word tagged. The POS tagger takes each word in a sentence and uses a heuristic method to find out which part-of-speech the word is, such as noun, verb or adjective. This data is useful when we want to reduce the number of potential word meanings in the disambiguation step. We will only get the synsets of that part-of-speech. If the word *bear* is a noun, we only get the nouns such as the animal, not the verb as in *bear a resemblance*. Since it is a heuristic method, the POS tagger may not be correct, in which case we get only the synsets of the wrong POS. The reason we have to tag the words so early in the algorithm is because the POS tagger needs the whole text in order to provide the best suggestions. If we remove stop words, we lose words that may be important for estimating the correct part-of-speech.

The abstract of the percussion instrument *hurdy-gurdy* after processing and removal of stop-words and duplicate words:

> hurdy-gurdy, stringed, instrument, produces, sound, hand, crank-turned, rosined, wheel, rubbing, strings, functions, violin, bow, single, notes, played, similar, melodies, keyboard, presses, tangents, against, one, change, pitch, acoustic, instruments, board, hollow, cavity, vibration, audible, hurdy-gurdies, multiple, drone, constant, accompaniment, melody, resulting, bag-pipes

Procedure 1 Process a Wikipedia abstract to prepare words for WordNet queries

Input: wikipediaAbstract,
Output: processedWikipediaAbstract, wikipediaAbstractWithoutStopwords

procedure PROCESSWIKIPEDIAABSTRACT
  for *wikipediaAbstract* **do**
    Remove extra newline
    Replace line breaks with whitespace characters
    Split words separated by forward slash
    Remove   to avoid the number 160 appearing in the text
    Remove [] and the containing text (e.g reference tags)
    Remove parenthesis and the containing text
    Remove forward-slash and the containing text (e.g phonetics)
    Remove - at the end of words
    Remove d′ at the start of French words
    Remove l′ at the start of French words
  wikipediaAbstractWithoutStopwords ← *wikipediaAbstract*
  for *wikipediaAbstractWithoutStopwords* **do**
    Remove ′s at the end of words
    Remove all apostrophes
    Remove symbols !@#$%& ∗ ()? <> +ˆ:;”, .′♭♯ = \\`_˜£€
    Remove words that has numbers in it
    Change to lower case (WordNet requires lowercase string)
    Remove stop-words
    Remove duplicate words
  return [*wikipediaAbstract*, *wikipediaAbstractWithoutStopwords*]

5.1.3 Identifying noun phrases

When analysing a text for useful properties, some of them may be described by multiple words or a noun phrase. Take for example the two-worded property *sustaining pedal*, which is one of the pedals on a piano that lifts the dampers from the strings to let them continue vibrating. The word sustaining and pedal can not sufficiently describe the sustaining pedal on their own. The meaning of the word sustaining depends on the word pedal, and it may not be clear if the word pedal is a sustaining pedal, a soft pedal or any other pedal unrelated to pianos. Therefore identifying noun phrases can lead to finding more useful properties in the text. We can identify noun phrases by using the Stanford parser which takes a sentence and label the noun phrases with a NP tag. We can then remove unnecessary parts of the noun phrase such as determinants. If the noun phrase contains more than two words, we can recursively check for other noun phrases after removing the first word. For example in the noun phrase *electric five-string bass* there is also the noun phrase *five-string bass*, and *bass*. The Stanford parser breaks sentences into phrases annotated by their type, such as noun phrase or verb phrase. Each phrase has parentheses for each word, tagged with their POS. For instance the sentence *Bob has a five-string bass guitar*, is given the output:

```
(ROOT
(S
(NP (NNP Bob))
(VP (VBZ has)
(NP (DT an) (JJ five-string) (NN bass) (NN guitar)))))
```

See Procedure 2 for how we extract the longest possible noun phrase from a line returned by the Stanford parser. Following this method we also find possible shorter noun phrases in the returned noun phrase using a simple recursive method where we remove the first word.

5.1.4 Synsets

WordNet provides a set of synsets for each word in the database, and each of these synsets has a unique definition. If we search for the word *string*, WordNet returns a set of 10 nouns and 7 verbs. We want to search WordNet for every word we found in the abstract and find the correct synsets so that we have a list of unambiguous properties representing the concept. WordNet also store the POS of a word, so we can use the POS tagger data to remove

Procedure 2 Get noun phrase from a line

Input: line,
Output: nounPhrase

```
procedure GetLongestNounPhraseFromParserLine
    if line is not a tagged as noun phrase then return null
    for each parenthesis in line do
        tags.add(contents of parenthesis)
    if tags.length < 2 then return null
    for each tag in tags do
        if tag is a symbol then continue
        if tag is an possessive ending then continue
        if tag is a determinant then continue
        nounPhrase ← nounPhrase concatenated with tag.taggedWord
    if nounPhrase word length < 2 then return null
    return nounPhrase
```

TABLE 5.1: We found the following noun phrases candidates in the *hurdy-gurdy* abstract

Hurdy-gurdy noun phrase candidates
stringed instrument
wheel functions
single notes
small wedges
their pitch
most other acoustic stringed instruments
other acoustic stringed instruments
acoustic stringed instruments
stringed instruments
sound board
Most hurdy-gurdies
multiple drone strings
drone strings
constant pitch

every synset of the wrong POS. For instance if we have tagged *piano* as a noun, we can remove the adjective *piano* which means soft.

5.1.5 Stemming

Sometimes WordNet gives no results because the word is in a different variant than the one in the WordNet database. For instance, if the word is *guitars*, it does not match *guitar* in the database. To return *guitar*, we can use a stemming algorithm to find the stem of the word. Often the stem is found in WordNet. This method is not always perfect though. Sometimes the stemmer can over-stem, cutting too much of the word leading to a word that has a different meaning.

5.1.6 Word sense disambiguation

Now that we have the key words and noun phrases extracted from the abstract, we will try to find the correct meaning of each word. Let's say we want to find the synset corresponding to the *bow* used to play the *cello*. How can we do this? We have to get a list of words that we will look for in each definition of *bow*. We call this list *scoring words*. *Scoring words* will consist of each word in the processed abstract plus a set of musical context words that can act as the musical context if the abstract lack sufficient musical words. Most likely the definitions we are looking for are related to a musical context. We use the algorithm in Procedure 3. It is based on the *Simple Lesk* algorithm we introduced in chapter 2.
The *context words* we use in this example are:

music, instrument, tone, tones, sounds, sounding, rhythm, melody, drum, flute, percussion, idiophone, bass, baritone, tenor, alto, soprano, note, pitch, audio, auditory, tune, tuning, acoustic

The *scoring words* combines these with the words of the *cello* abstract:

cello, violoncello, bowed, plucked, string, instrument, four, strings, tuned, perfect, fifths, low, octave, lower, viola, violin, family, musical, instruments, includes, double, bass, solo, chamber, music, ensembles, orchestras, section, symphony, types, rock, bands, second-largest, lowest, modern, orchestra, largest, pitch

We will remove duplicates of words that are in both lists so that each word is equally influential on the scoring. Duplicate words are:

music, instrument, bass, pitch

To add score to a definition, we count the number of words (excluding stop-words) in each synset definition which also exists in the *scoring words*. The first definition of *bow* is:

a knot with two loops and loose ends; used to tie shoelaces

None of these words are in the *scoring words* of a cello, therefore we give it a score of 0. The next definition is:

a slightly curved piece of resilient wood with taut horsehair strands; used in playing certain stringed instruments

One word, instrument, is found in the cello scoring words, therefore we give it a score of 1. We select the synset that has the highest score, which in this case was the correct bow definition. This approach is heuristic and will not always find the correct synset. Sometimes the word is similar to, but not equal to the scoring word. For example if the definition had the word *bow*, and scoring words had *bowed*, we would not find it. One way to match these would be to search for *bow* in the start of the string *bowed*, and also search for *bowed* in the start of *bow*. Now we would get matches when the scoring word starts with the definition word and vice versa. An alternative way could be to use a stemming algorithm on the words and see if either the scoring word or scoring word stem matches the definition word or definition word stem.

An alternative additional scoring system could be to use *context synsets*, specific synsets that automatically give an extra point for being relevant to the category. For example the musical synset of the word *instrument* would automatically get an extra point over non-musical synsets of instruments like the more general synset *device*.

Other techniques could be to give more score to certain words or synsets than others. Some words may be more important, such as the concept name itself. One of the definitions of *hammer* include the word *piano*, which is clearly an important word if this is also the instrument. Therefore it could get a score of 5 when other scoring words get 1 point.

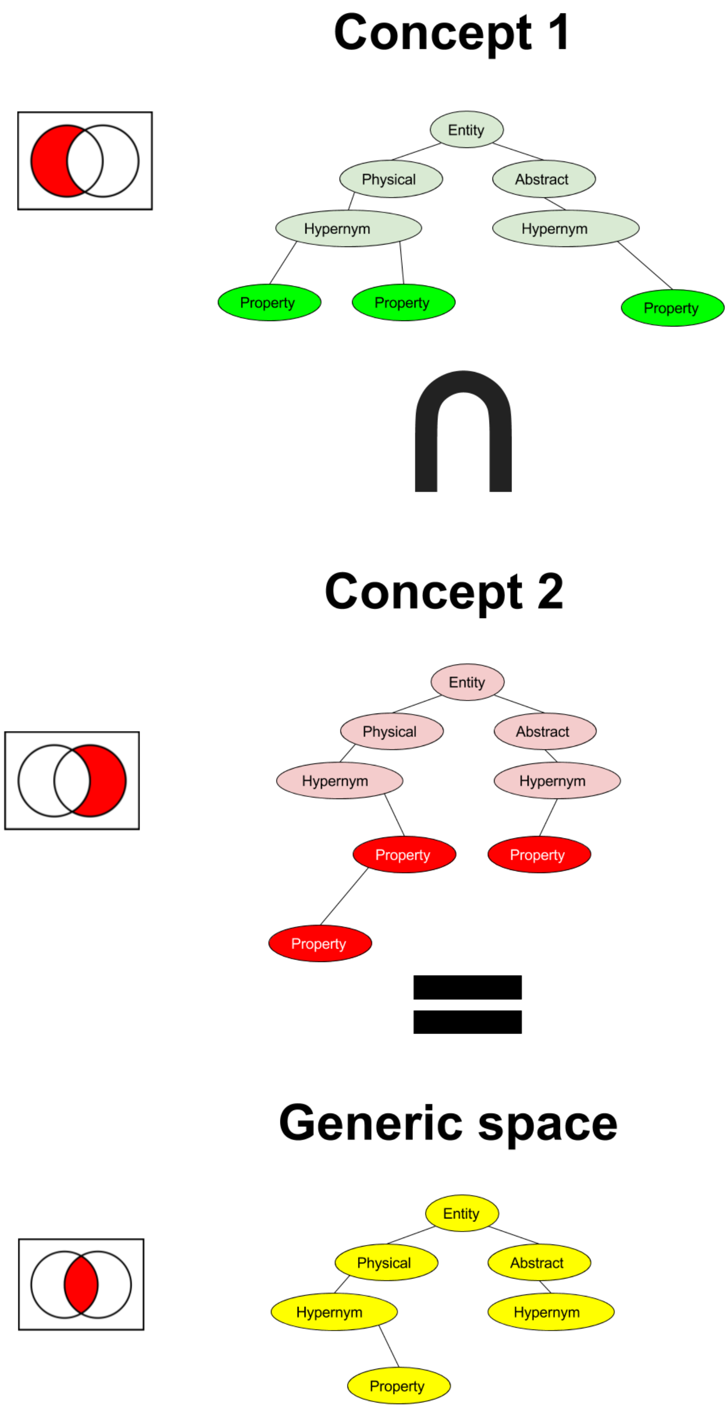

FIGURE 5.3: The generic space is created by the intersection of properties (strong color) and hypernyms (weak color) of the two concepts. Since every node in the generic space has subtrees in both concepts, we can use it to find properties correlating with a given property. The most general word *entity* can be seen branching vertically into specific categories.

Procedure 3 Get the definition which is most likely to fit the word

Input: ambigousWord, definitions, scoringWords
Output: highest scoring definition

procedure GETMOSTLIKELYDEFINITION
    for each *definition* in *definitions* **do**
        definitionScore $\leftarrow 0$
        for each *definitionWord* in *definition* **do**
            for each *scoringWord* in *scoringWords* **do**
                if *scoringWord* matches *ambigousWord* **then**
                    continue
                if *scoringWord* is the first part of *definitionWord* **or** *definitionWord* is the first part of *scoringWord* **then**
                    definitionScore $\leftarrow$ *definitionScore* $+ 1$
        if *definitionScore* $>$ *definitionMaxScore* **then**
            definitionMaxScore $\leftarrow$ *definitionScore*
            topDefinition $\leftarrow$ *definition*
    if there is no *topDefinition* **then return** null
    return *topDefinition*

5.2 Generic space

5.2.1 Generalization using hypernyms

Our approach to concept blending is inspired by the use of amalgams. When using WordNet we can easily generalize since we have access to each synsets tree of hypernyms. So if we consider the concepts *a red French vehicle* and *a German minivan*, we can find that *French* and *German* share the hypernym *nation*, and minivan and vehicle share the hypernym *vehicle*.

The point of the algorithm is to find properties in one concept that has correlating properties in the other concept. If the other concept has a property in the same category, there may be a possibility for the property to fit into the other concept.

Our solution give score proportional to the speciality of the word and penalize proportionally to how general the generalization is, aiming for the most specific generalization. In the example above, minivan should get a good score, since it is specific and the other concept is a vehicle, which makes not a too general generalization. However a minivan and SUV would give a better score, since they are both cars, which is less general than vehicle.

The algorithm we use to create the generic space between is shown in Procedure 4. We can pass *piano* as the source concept and *hurdy-gurdy* as the target concept. After processing the abstracts and finding the most likely synsets of each property, we can use WordNet to retrieve a list of every hypernym ancestor of each synset, which will go from the least general to the most general generalization. They can be accessed from the array *concept.property.hypernyms*. To create the generic space, we simply retrieve all synsets and hypernym synsets in each concept and use a intersection set operation to keep what we find in both concepts.

Procedure 4 Get all synsets in the generic space between two concepts

Input: sourceConcept, targetConcept
Output: highest scoring definition

```
procedure GetGenericSpace
    for each property in sourceConcept.properties do
        sourceSynsets ← sourceSynsets + property
        for each hypernym in property.hypernyms do
            sourceSynsets ← sourceSynsets + hypernym
    for each property in targetConcept.properties do
        targetSynsets ← targetSynsets + property
        for each hypernym in property.hypernyms do
            targetSynsets ← targetSynsets + hypernym
    sourceSynsets ← removeDuplicates(sourceSynsets)
    targetSynsets ← removeDuplicates(targetSynsets)
    return SetOperations.intersection(sourceSynsets, targetSynsets)
```

5.2.2 Finding correlating properties

In order to provide good suggestions, we need to find properties that share a hypernym in the generic space with some property in the other concept. For instance a football field and a golf terrain share a common hypernym *piece of land* which means they form a correlation and could therefore be more compatible for blending than less related pairs.

5.3 Finding good blending suggestions

In our application we used a solution where properties from each concept were sorted with a score that represents how well they are suited for blending. We were improving the algorithm by identifying patterns typical of good candidates in the tree structure of the property words and their hypernyms.

5.3.1 Depth of the word in hypernym tree

The first pattern we found was that words that have high generality are not useful. We want more specific words like *drumstick* rather than general and vague words like *object*. Therefore we gave points equal to the depth of the word in the tree.

5.3.2 Branch length to shared hypernyms

The second pattern we found was when properties of two concepts share a common root or hypernym, the length of the branch from this root could give us useful information on how specialized the property was. The longer the branch length, the worse the candidate seemed to be. Therefore we punished the word by subtracting points equal to the branch length.

5.3.3 Penalizing certain categories

With these criteria, our winning words were specific, but often abstract. A complete instrument or a person (e.g. the inventor) were not interesting blending suggestions. Therefore we chose to punish the words significantly when being in the branch of *abstract entities*, *instruments* or *individuals*. The resulting suggestions were then both physical and specific, which seems to be optimal for creating new instruments.

5.3.4 Removing properties in penalized categories

An observation of the properties that connected in the general space, showed that a lot of these were in penalized categories. For example a property of the penalized category instrument is a device, which makes devices in the other concept score higher than if there were no other devices in the first concept. A way to avoid this could be to remove properties of penalized categories altogether. This leads to far less blending suggestions, but the suggestions that are left are of relatively high quality. Since no blending suggestions are in penalized categories also makes the penalty score redundant.

5.3.5 Ranking Compatible Blending Suggestions

First we find each property that are not in the generic space themselves but has a hypernym in the generic space. This means that this property can be blended and we can give it a compatibility score for whether it is a good blending suggestion. We refer the branch from the synset and hypernyms subsuming its hypernyms in the generic space for the blending *subtree*. We set the compatibility score to the depth of its least general hypernym in the generic space minus the length of the *subtree*. Finally we can optionally subtract a constant penalty value if the property is or has a hypernym in a penalized category. With this score we can store the unique properties in a list which we can rank using a heuristic function. The algorithm for finding unique properties and ranking them is shown below in Procedure 5.

Procedure 5 Perform creative strategy by ranking a concepts subtrees by finding a pair in the other concept with the highest compatibility, defined by the expression Depth - BranchLength - Penalty

Input: properties, concept, genericSpace, penaltyCategories, penaltyPoints
Output: a list of branches ranked by their blending compatibility score

```
procedure RankCompatibleBlendingSuggestions
    for each property in properties do
        if property.name = concept.name
           or property has not been disambiguated
           or property is in genericSpace then
            continue
        for i = 0 to property.hypernyms.length do
            if property.hypernyms[i] is not in genericSpace then
                continue
            subtree.branch ← [property] concatinated with property.hypernyms[0:i]
            depth ← property.hypernyms.length − i
            branchLength ← subtree.branch.length
            if subtree.branch has an element in penaltyCategories then
                penalty ← penaltyPoints
            subtree.compatibility ← depth − branchLength − penalty
            subtrees ← subtrees concatinated with subtree
            break inner for loop
    return subtrees sorted by subtree.compability in descending order
```

5.3.6 Completion and elaboration of blend

After ranking the blending suggestions in each concept, we are left with the problem of completing and elaborating the blend. The least radical blending would be to move the element with the highest score over to the other concept. When blending *guitar* and *violin*, the highest score our implementation gives to an element is the *bow* in violin. Therefore we can move the bow over to guitar, and we can assume that the bow will be used on the guitar in a similar manner to how it is used on a violin. We have not represented any other relation between bow and violin than the fact that *violin has bow*. And now *guitar has bow*. It will require human interpretation of the original description to further understand how it is used as our solution does not do any further interpretation of the elements and their relation to the concept.

Another method for blending the suggestions can be to have a compatibility threshold, so that all elements scoring higher than the threshold are moved to the other concept. You can then look at either two blended concepts, or the concept with the highest sum of scores could be selected as the blend space. The compatibility score can be positive or negative, depending on whether the depth value is higher than the branch length or vice versa. The ideal threshold may be hard to define, it would depend on whether the number of hypernyms is a good measure of generality. In some cases there may be many hypernyms in WordNet to describe a short distance in generality. Some things may have more categories than others. For example, the chess pieces have a lot of hypernyms since they are among diverse types of object, but if the chess piece *rook* were the only gaming equipment to exist, we would not need hypernyms such as *chess piece* or *board game piece*. Some objects will probably diversify as time go by and new inventions appear. Therefore the number of hypernyms may not be a good measure of generality, especially when comparing over large distances in the category tree. But it may be enough to tell us a somewhat vague indication of the generality which could be useful even though it is not perfect. Further research should be done on how to achieve a better measure of generality.

Chapter 6

Results

The main output of the application is a representation of the final blend where a blending suggestion has been moved to the other concept. We use a graph to visualize this new blending and the parameters used to calculate the score. The blended tree would be too large to display in a graph, therefore we show the branch where the new property and its correlating properties are located. The ancestor categories of the shared hypernym are cut off, but we display the distance from this node to the most general word in the tree, *entity*, as a depth value.

6.1 Identifying right criteria for ranking blends

6.1.1 Speciality

We first started by blending musical instruments without any ranking parameters. The blending suggestions we got, were of varying quality. We recognized that the best suggestions were the ones that were most specific, such as the ones that were found due to being in the category *device* or *artifact*. Therefore we introduced two criteria to give a higher score to more specific suggestions. The first was to give a positive score based on the depth of the shared hypernym of the blending suggestion and its correlating properties in the other instrument. For example if we blend *piano* and *hurdy gurdy*, it gave a high score to the suggestion of transferring the *piano chamber* over, since the hurdy gurdy also has a *bodily cavity* which is the hypernym of chamber. The bodily cavity had a depth score of 7, which means that it has 6 ancestor categories that are more general. Therefore it is not very general.

The other criteria we introduced was a score negatively proportional to the branch length. From piano chamber, it is only one hypernym up to bodily cavity. This tells us that the suggestion is not very far from its shared category in speciality. Therefore the score given by the branch length criteria was only

-1. The suggestion was then given the total score of 7 minus 1, which is 6. The properties used in the calculation is visualized in Figure 6.1.

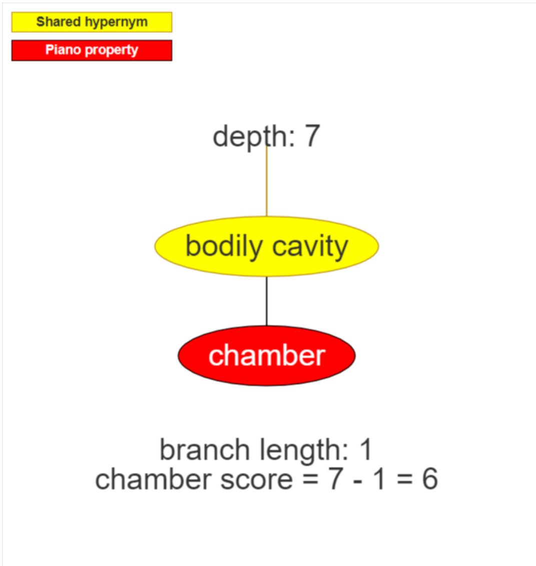

FIGURE 6.1: One of the suggestions when blending piano and hurdy gurdy. It suggests to move *chamber* from piano to hurdy gurdy, since it is a type of *bodily cavity* which is also one of the properties of hurdy gurdy, thereby forming a correlation.

6.1.2 Category penalty

We observed that a lot of suggestions were not useful so we took note of the categories bad blends shared. Different instruments were often mentioned in the abstracts, which were then interpreted as properties. It does not make sense to transfer whole instruments to other instruments. It would be too invasive since concept blending are more about blending parts, not adding whole concepts to others. Therefore we introduced a penalty score for the category *instruments*. Later we also decided to remove properties of the category instruments altogether since they would still form correlations with other devices, since device is one of the hypernyms of instrument.

We also observed that properties of the category *individuals* were not useful as well, since they were typically people like the inventor of the instrument or a famous musician using this instrument. It does not make sense to blend the people involved, since they are not parts or properties of the instrument. What use would there be to transfer an inventor over to an instrument they have not invented?

We initially thought that abstract entities was a bad category to blend, since good blending examples are typically physical entities. But on further inspection, we found that some of these entities were useful, like shapes and other attributes. Instead, we penalized certain abstract entities that we were more certain being bad categories. *Psychological feature* are probably not interesting if they most likely are part of something other than the instrument, since instruments does not have a psychological quality. We also penalized *communication*, since we are not interested in the message, language or style expressed by the instrument. This is usually a matter of choice from the musician and not the instrument. Although some instruments are typical for a certain style, we do not want to define the style of new instrument in advance of its use. *Body parts* are not something we want to blend as parts, but in future approaches that look for relations, it would be useful to describe relations like *plucked by fingers*.

6.1.3 Number of correlating properties

We observed that in some cases the top blending suggestions got an equal score. Could we find a difference in quality between these? We introduced another parameter, the number of correlating properties in the other concept. If the other concept has a lot of properties in the same shared category as the blending suggestion, this may tell us that the property is more compatible than the one with few correlations. The property may fit with the other concept in more than one way.

6.1.4 Introducing factors for each parameter

We realized that a factor of one for each parameter may not be yield the best results. Therefore we introduced a factor for each parameter so that they could be tweaked to push blends of higher quality to the top. Then we could find an optimal range of values for pushing the best suggestions to the top.

6.1.5 Evaluation of parameters

The parameter which had the most obvious influence on the quality of the blending suggestions was the depth of the shared hypernym. This made general properties score low. The levels of categories between specific and general hypernyms seem to vary in WordNet, so it is not a perfect measure of speciality. Special hypernyms may branch into several niche categories affecting the quality of this measurement, while the top of the tree keep the amount of generality more evenly over the same score. The branch length parameter could penalize some suggestions which were too specialized compared to the shared hypernym. It did not seem to be as vital as the depth value however. We compared blends that had the same depth and branch length, by looking at the number of correlating properties. We wanted to see if there was a trend where a large or small number were better. However it did not seem to not matter significantly.

6.1.6 Other potential criteria

We thought of other possible parameters that could be used, that we did not implement and evaluate:

- The branch lengths of the correlating properties could be counted or averaged to give an indication of their speciality.
- The difference between the branch lengths of the suggested property and its correlating properties could give an indication of compatibility. For example we may not want to blend a property with a branch length of 5 when the correlating properties are very general in comparison, like if it has a length of 1. Blending *game equipment* like the special chess piece *rook* just because the other concept had a more general game equipment like *game board*, may not be as good as if it were type of board with similar level of specificity like *backgammon board*.

6.2 Blending musical instruments

Here are a couple of blending examples when using the application to blend musical instruments. We blended a variety of instruments types in order to get an indication of how the algorithm performed in different scenarios. First we present the retrieved properties and blending suggestions when we use an unaltered abstract as it was extracted directly from Wikipedia. Then we show an upgraded blending using an improved abstract we created manually by removing unhelpful sentences and adding new useful facts. If synsets were wrong, we also manually corrected them. This way we could evaluate the second and third step of the application in an optimal environment. Properties not included in the presentation are those where a meaning could not be found, or was of a penalized category. The properties where the wrong meaning was selected are still included, since this is the way the application would use the properties when unaided by a user.

6.2.1 Blending banjo and melodica

Unaltered abstract

The banjo abstract the application retrieved from Wikipedia, was not very detailed in its physical descriptions.

Retrieved properties of banjo: (noun phrases in quotation marks)

five, membrane, frame, resonator, called, head, animal, skin, circular, forms, africans, "animal skin"

By looking at these properties we can hardly recollect the main parts of the instrument, although some important words like *resonator* and *membrane* are included.

Melodica properties:

key, pump, keyboard, top, played, blowing, mouthpiece, fits, hole, side, pressing, reed, popular, asia, modern, form, italy, century

These words are quite good, we have essential parts like *keyboard*, *mouthpiece* and *reed*. The word *pump* is not used in a melodica however, and we can see some uninteresting cultural words.

The top blending suggestion we got was to move the word *five* from banjo to melodica with the score of 4, correlating with the property *century* through the common category *integer*. *Century* is a part of the cultural description of melodica, rather than the more useful sentences describing physical attributes.

Optimal environment

By improving the banjo abstract we got more relevant properties like *body*, *strings* and *frets*, and a variety of materials which the banjo parts and strings are made of.

Improved banjo properties:

four, five, six, stringed, thin, membrane, stretched, frame, cavity, resonator, called, head, plastic, animal, skin, circular, body, consists, rim, wood, metal, tensioned, similar, tone, ring, assembly, project, sound, tuned, friction, tuning, peg, gear, worm, machine, frets, standard, played, strung, strings, string, wound, steel, alloy, nylon, gut, achieve, old-time, separate, plate, pot, forward, volume

In the new set of melodica properties we got rid of cultural words like *Asia*, *Italy* and *century*, and retrieved more relevant ones like the materials *plastic* and *wood*.

Improved melodica properties:

keyboard, top, played, blowing, air, mouthpiece, fits, hole, pressing, key, flow, reed, two, three, light, portable, majority, plastic, wood

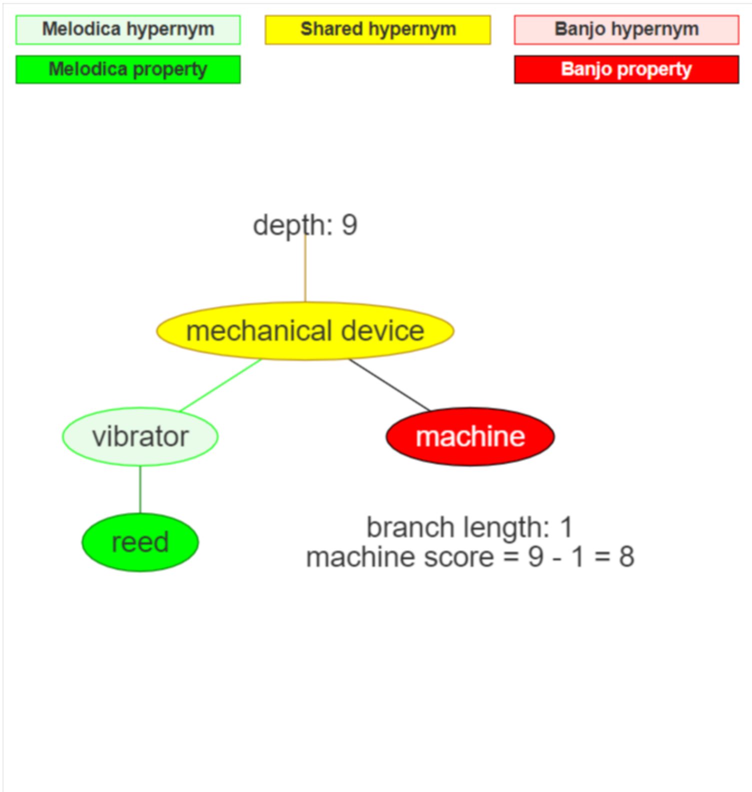

FIGURE 6.2: The top suggestion when blending banjo and melodica using improved abstracts. It suggests to move top property *machine* (in red) as in *worm gear machine head* from banjo to melodica.

The top blending suggestion we got was introducing *machine* as in the guitar's worm gear machine head for tuning the strings (see Figure 6.3). It correlated with the *reed* of melodica, through the category *mechanical device*. It is more specific than the other categories, but still so general that they are not closely related in function. Although we found the noun phrase *machine head*, it did not exist in WordNet, giving us an indication of the limits of its database. The user may be inspired to create a melodica where you can tune

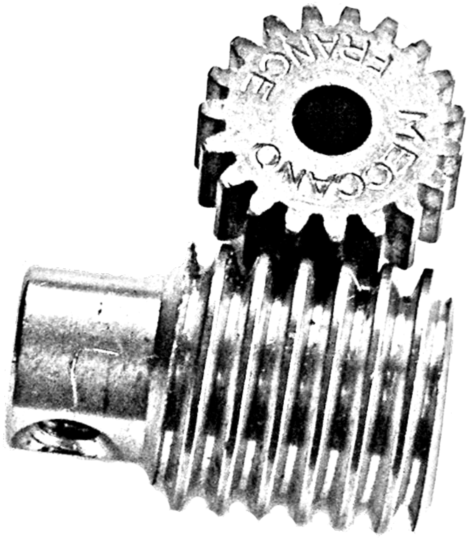

FIGURE 6.3: A worm gear which is used in a machine head for tuning the strings of banjos and guitars

the reeds with a similar tuning device as guitar. The calculation of this blending suggestion is visualized in Figure 6.2.

6.2.2 Blending harp and violin

Unaltered abstract

The physical description in the harp abstract was very short, and important properties like *strings* was misinterpreted by the word sense disambiguation as stringed instruments and since whole instruments are left out.

Harp properties:

stringed, musical, soundboard, africa, ages, renaissance, family, near, played, burma, utilized, modern, "Latin America", "Near East", "modern era"

Of these properties only the noun *soundboard* was useful for the application. There was also a lot of uninteresting cultural words.

Violin properties:

family, regular, typically, perfect, commonly, played, bow, fingers, prominent, classical, country, electric, forms, rock, iranian, sometimes, called, fiddle, particularly, irish, traditional, regardless, italy, europe, stradivari, guarneri, amati, century, brescia, cremona, austria, reputation, quality, sound, disputed, hands famous, mass-produced, commercial, cottage, saxony, bohemia, formerly, sold, sears, roebuck, co, mass, "violin family"

Here we also have a lot of words deriving from cultural descriptions. Other than *bow*, the words were rather uninteresting.

Due to the low quality of the properties retrieved, the blending suggestions we got was not surprisingly getting low scores. The top blending suggestion was to move the word *cottage* from violin to harp. The irrelevant word cottage comes from a sentence explaining that violins are often made by cottage industries. It correlated with another *construction*, the *soundboard* of a harp. Moving the soundboard back to violin was also the top suggestion of properties to move from harp, but with a lower score since it had a longer branch to the shared property *construction*. The violin *bow* got a low score, due to the lack of *sticks*, *implements* or *instrumentalities* in the harp properties. It correlated with *soundboard*, but these three categories between the bow and the shared category *artifact* drove the score down.

Optimal environment

After using improved abstracts we got rid of a lot of cultural words, and with important words like *strings*, pillar and *pedals*, we could get a better image of the harp.

Harp properties:

stringed, strings, angle, soundboard, plucked, size, played, larger, heavy, rest, floor, catgut, nylon, metal, combination, neck, resonator, frame, pillar, support, arch, bow, modern, extend, range, adjusting, levers, pedals, modify, pitch

The changes lead to the violin getting new valuable properties like *string*, *fingerboard* and *sound hole*.

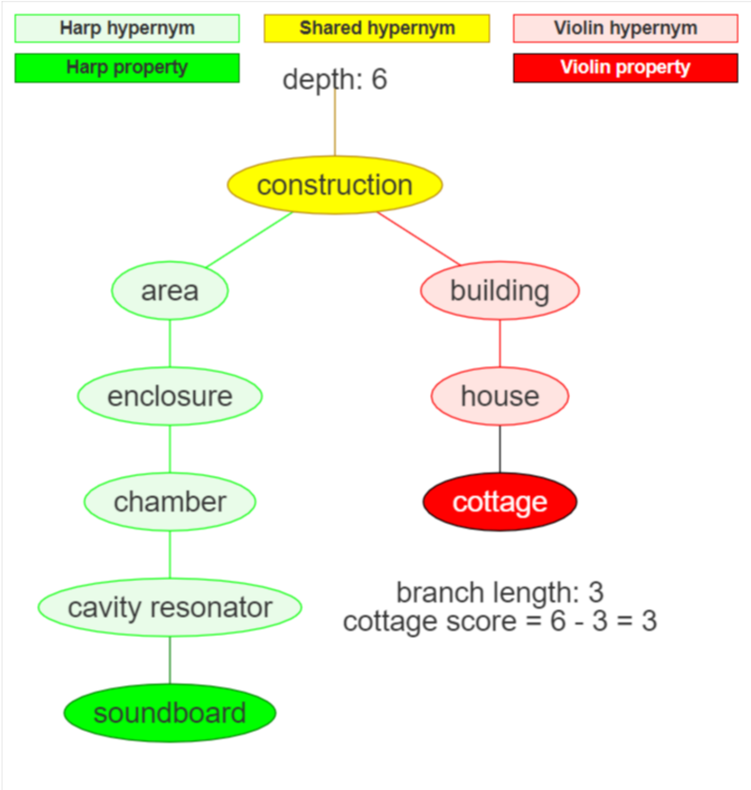

FIGURE 6.4: Top suggestion when blending harp and violin. It suggests to move top property *cottage* (in red) from violin to harp

Violin properties:

wooden, string, family, scroll, neck, fingerboard, bridge, sound, hole, tail-piece, four, strings, tuned, commonly, played, drawing, bow, plucking, called, wood, strung, gut, synthetic, steel, "sound hole"

We got two top blending suggestions with the score 5. The first was to blend the violin *bow* with harp, correlating through the category *implement* with the harp *pedals* that bend the strings. Harpists have played harps with bows, but since you have to insert the bow between harp strings, it is not easy to change notes. However since the harpists have two available hands, they can use two bows. The calculation is visualized in Figure 6.5.

The top suggestion to move from harp to violin was *pedals* which modify pitch, correlating with the violin *bow* through the category *implement*. This could be an interesting feature, but since the violin has no frets, modifying pitch is already possible with the glissando technique where you slide fingers up or down the string. However if you want to play lower than the open string or bend further than the fingers can reach (unlikely on a small violin), a pedal could be useful.

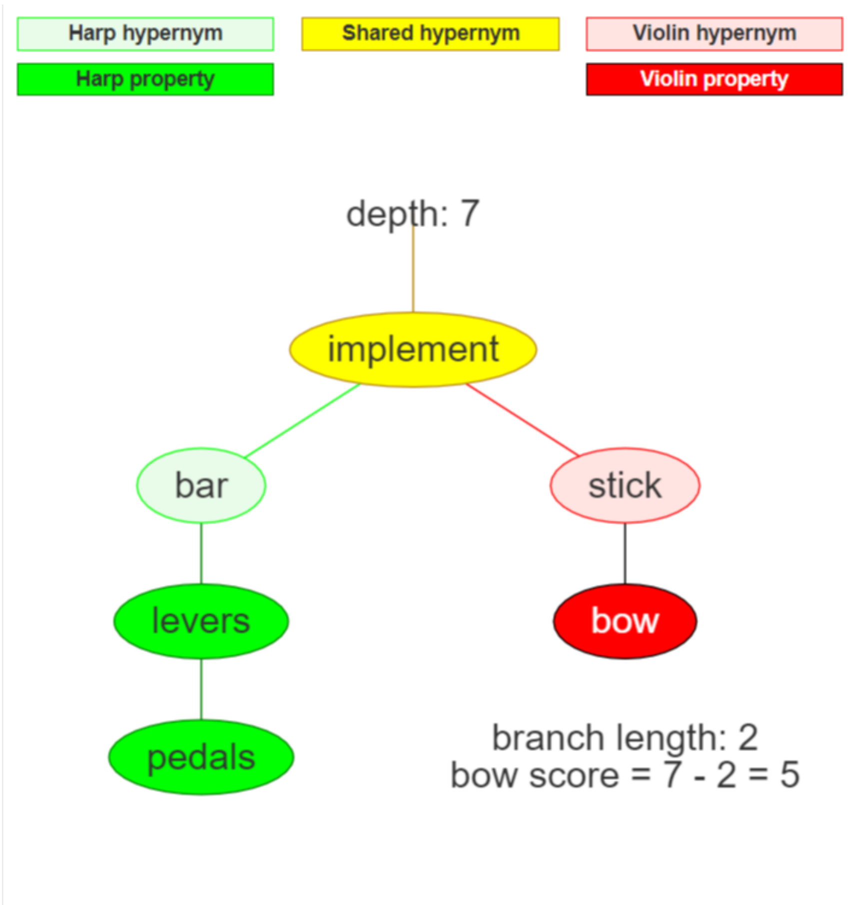

FIGURE 6.5: The top suggestion when blending harp and violin when using manually improved abstracts as a data source. It suggests to move top property *bow* (in red) from violin to harp.

6.2.3 Blending dobro and harp

Unaltered abstract

Dobro properties:

word popular term resonator brand currently gibson inverted biscuit national steel solid body

Harp properties:

stringed soundboard plucked africa popularity ages renaissance near played myanmar utilized modern era Latin America Near East modern era

We started with the unaltered abstract. With these properties we got similar blending suggestions from both dobro and harp. It suggested to move *body*, which is a *cavity resonator*, from dobro to the harp, while it also suggested to move the *soundboard*, also a cavity resonator, from harp to dobro. Being both one level below their shared category, they got the same score. Although a dobro also had a body, this word came from a reference to a solid body guitar, which is not the same. Other suggestions were not nearly as good in terms of score. Moving *steel* from dobro to harp got as score of 1, but the wrong meaning was selected. The WSD chose the meaning of a steel sharpener, instead of the material. Manually improving the WSD of all properties did not improve the suggestions and scores significantly.

Optimal environment

After improving the abstracts by manually selecting useful parts from the Wikipedia abstract, we got a somewhat different result. Body was no longer a property, but *soundboard* from harp was still the best suggestion since dobro had a *cavity resonator*, which is its hypernym. The best suggestion of dobro was *aluminium* from its cast aluminium spider, linking with the *metal* that strings may be made of in a harp. Other suggestions that were less rewarded, but more interesting would have been the cone or bowl forming the resonator. An interesting but low scoring suggestion from harp is its pitch bending levers or pedals.

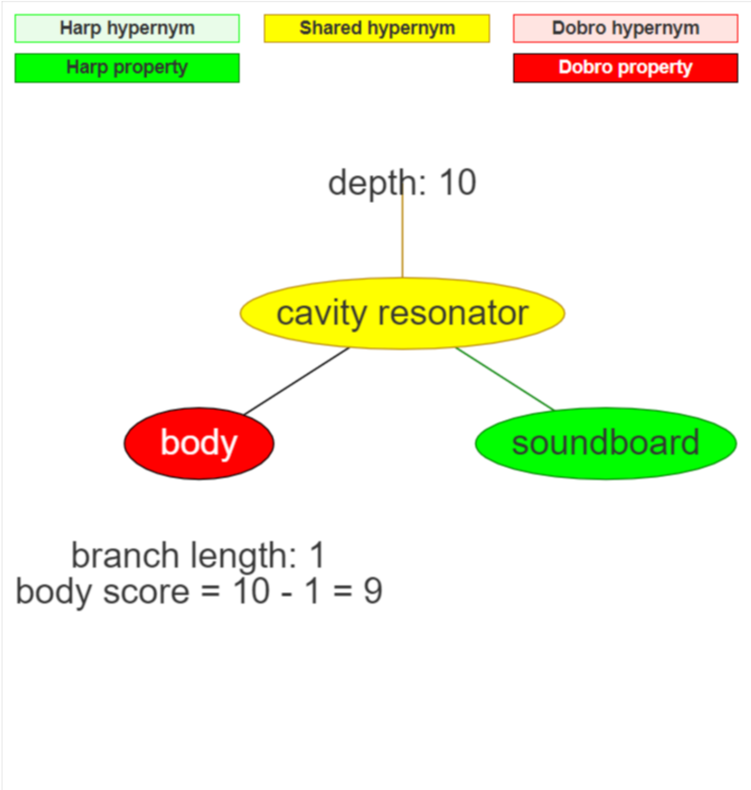

FIGURE 6.6: Top suggestion when blending dobro and harp. It suggests to move top property body (in red) from dobro to harp

6.2.4 Blending guitar and violin

Unaltered abstract

Interesting properties describing the guitar directly include *body* and *resonating chamber*. There was also some words describing various accessories and effects such as *pick*, *loudspeaker*, *equalizer* and *distortion*.

Guitar properties:

fretted, usually, six, sound, projected, acoustically, hollow, wooden, box, electrical, amplifier, speaker, typically, played, strumming, fingers, fingernails, hand, pick, fretting, left, type, traditionally, constructed, strung, steel, distinguished, modern, renaissance, baroque, three, types, acoustic, classical, sometimes, called, tone, body, acts, resonating, chamber, plucked, individually, opposed, strummed, country, united, low-pitched, below, regular, electric, loudspeaker, makes, loud, performers, audience, hear, signal, electronically, manipulate, shape, equalizer, huge, electronic, units, commonly, ones, distortion, employed, solid, eventually, "right hand", "left hand", "resonating chamber"

The violin properties were rather uninteresting, except for the *bow*.

Violin properties:

family, regular, typically, perfect, commonly, played, bow, fingers, prominent, classical, country, electric, forms, rock, iranian, sometimes, called, fiddle, particularly, irish, traditional, regardless, italy, europe, stradivari, guarneri, amati, century, brescia, cremona, austria, reputation, quality, sound, disputed, hands famous, mass-produced, commercial, cottage, saxony, bohemia, formerly, sold, sears, roebuck, co, mass, "violin family"

The top blending suggestion we got was to move the *pick* or plectrum used to pluck the guitar strings, over to violin. It correlated with the violin *bow*. This is meaningful, since both are used to vibrate the strings. This relation is however not mapped in the algorithm, but it gives us a glimpse of how a more optimal concept blending algorithm would operate. For example it could look for the words pick and string, and look for relations between these using information extraction methods on a large set of relevant documents. If the same relation exists with a correlating word of *pick*, such as bow, we may have a good suggestion for blending. It could also look for correlations with the word *string*. Then we could find other correlating objects that are vibrated. The relation *vibrate* is a verb that could be generalized using WordNet. It has one hypernym *move*, which the algorithm could use to find correlating objects that *move* a *string*.

Optimal environment

New and interesting properties include *string*, *frets*, *neck*, *bridge*, *soundboard* and *sound hole*.

Guitar properties:

fretted, usually, six, strings, key, frets, truss, rod, neck, body, transducer, bridge, soundboard, sound, hole, projected, acoustically, hollow, wooden, plastic, wood, box, electrical, amplifier, speaker, typically, played, strumming, plucking, pick, fretting, constructed, strung, gut, nylon, steel, distinguished, tone, acoustic, vibration, acts, resonating, chamber, comprehensive, string, plucked, individually, opposed, strummed, electric, loudspeaker, makes, loud, audience, hear, signal, electronically, manipulate, shape, equalizer, huge, electronic, units, commonly, ones, distortion, employed, solid, eventually, feedback, "electro-acoustic transducer", "sound hole", "resonating chamber"

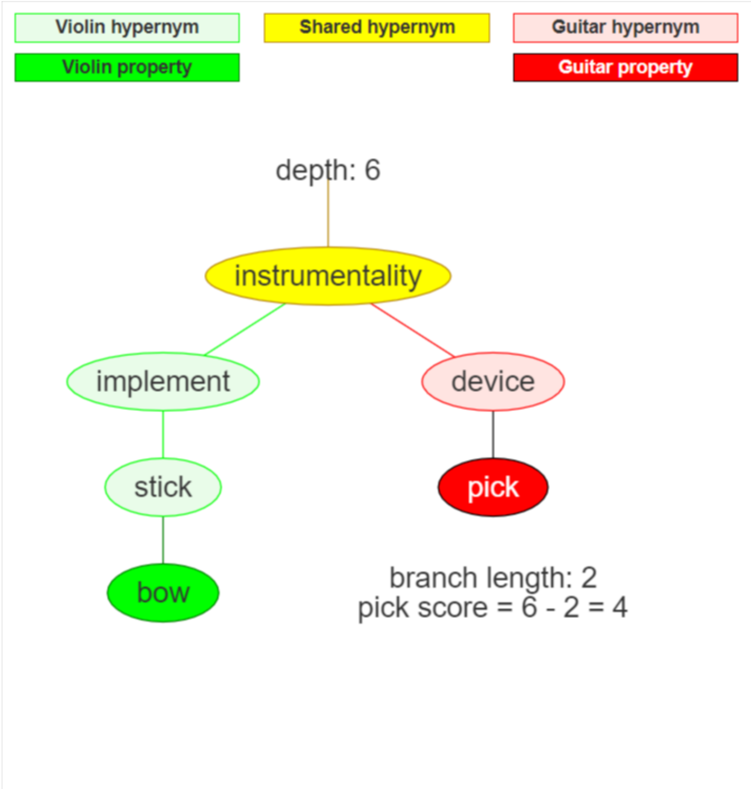

FIGURE 6.7: Top suggestion when blending guitar and violin. It suggests to move top property pick (in red) from guitar to violin

The improved violin abstract gave us new valuable properties like *string*, *fingerboard* and *sound hole*.

Violin properties:

wooden, string, family, scroll, neck, fingerboard, bridge, sound, hole, tail-piece, four, strings, tuned, commonly, played, drawing, bow, plucking, called, wood, strung, gut, synthetic, steel, "sound hole"

There was a lot of blending suggestions from guitar to violin that shared the top score of 6, and also one from violin to guitar. All of them had a depth of 7 and branch length of 1. It suggested to move the number *six* as in six strings from guitar due to a correlation with the number *four* in violin. And four strings were naturally suggested to move to guitar as well. These examples have been already been produced, although four string guitars are not very popular and six string violins are not practical to play. Like the unaltered abstract, moving pick from guitar to violin was also the top suggestion, but this time it got a higher score because it correlated with the property *bridge* through the more specific hypernym *device*. This underpin how the results are not affected by good relations between multiple properties.

Another top blending suggestion was to move a *rod*, as in the guitars truss rod, over to violin. It correlated with the violin *bow* since they are both implements. A truss rod is a steel bar inside the guitar neck, that keeps the thin neck from curving due to the tension of the strings. Therefore it is not closely related to a bow, but making a violin with a truss rod could be useful if we wanted to make the neck thinner or use string materials of higher tension than a normal neck would withstand.

Interesting suggestions that have lower score include moving frets from guitar, but it forms a less interesting correlation with *bow* through the hypernym implement. It would be desirable if the fretboard of guitar formed a correlation with the fingerboard of violin, but fretboard is not recognized by WordNet, although fingerboard is. The hypernym of fingerboard is *strip*, defined as a thin piece of wood or metal. Fingerboard could form interesting suggestions by blending violin with certain objects since people have been mounting strings on shovels and other such thin pieces of wood or metal. However the *bar* of a shovel is considered by WordNet as an instrumentality while *strip* is considered a building material, meaning they are separated too generally to achieve a good score. This is another example of how it is hard to use WordNet to find close relations by comparing only single synsets.

6.3 Alternative blending category

6.3.1 Sports

The algorithm was tested on the alternative blending category sports. The context words to help the scoring of synsets were:

sport, athletic, ball, field, team, goal, rules, foul, score, player, referee, match, racket, club, net, skates, skis, helmet, pads, bat, pitch, court, tee, green, win, draw, loss

The penalized category was ’sport’. We would not want sports to be suggested, since it is the category itself, and sports as a whole are not interesting to blend.

6.3.2 Blending soccer and golf

Unaltered abstract

The abstracts of soccer and golf were quite good, containing several important facts. The interesting properties we got from the soccer abstract was *eleven*, *team*, *players* and *goal*. Words like *ball* and *field* had the wrong meaning, and ball was removed because it was interpreted as a sport.

Soccer properties:

association, commonly, team, played, two, teams, eleven, players, spherical, million, dependencies, game, rectangular, field, goal, object, score, getting, opposing, allowed, touch, hands, arms, play, mainly, strike, pass, body, except, scores, goals, match, wins, draw, declared, time, penalty, shootout, competition, laws, codified, england, governed, internationally, international, cups, four, extra, time, "World Cups"

Golf properties that were well retrieved included *players*, *cup*, *terrain*, *fairway*, *trap* and *hazards*. Unfortunately *club* and *ball* were interpreted as a basketball club and a ball game. *Holes* were interpreted as one period of golf, instead of the physical hole. Noun phrases such as *putting green* and *sand traps* indicate that WordNet has quite a bit of golf terminology in its database.

Golf properties:

golf, club, players, various, clubs, hit, series, holes, course, strokes, unlike, games, standardized, playing, terrains, key, game, level, played, arranged, progression, hole, contain, tee, box, start, putting, containing, cup, standard, forms, terrain, fairway, traps, hazards, specific, layout, arrangement, "playing area", "putting green", "sand traps"

The top blending suggestion we got was blending moving *goal* from soccer to golf. However it formed a correlation with the word game, interpreted wrongly as a game equipment. The blending suggestions were riddled with bad correlations such as this, and it was obvious that we had to improve the WSD to get good results. The abstracts however were quite good.

Optimal environment

The new properties we got when using improved abstracts and manual WSD include *ball* and *field*.

Soccer properties:

team, played, two, teams, eleven, players, spherical, ball, game, field, goal, object, score, getting, opposing, allowed, touch, hands, arms, play, penalty, strike, pass, body, except, scores, goals, match, wins, level, draw, extra, time, shootout, format, competition, formation, player, roles, include, strikers, forwards, midfielders, boundary, lines, markings, define, position, kicks, corner, balls, "extra time"

Important new golf properties were found, such as *club*, *ball*, *hole* and *green*.

Golf properties:

club, ball, players, various, clubs, hit, balls, series, holes, course, strokes, games, standardized, playing, varied, terrains, key, game, level, played, progression, usually, hole, contain, tee, box, start, putting, green, containing, cup, standard, forms, terrain, fairway, rough, grass, sand, trap, bunkers, hazards, water, rocks, fescue, specific, layout, arrangement, lowest, individual, stroke, play, score, complete, round, team, match, teeing, ground, set, two, markers, bounds, legal, surrounded, pin, normally, variety, iron, wedge, wood, "playing area", "putting green", "sand trap", "teeing ground", "putting green"

When performing the algorithm in the improved environment, it still suggested to move *goal* from *soccer* to *golf*, with a depth of 8 minus branch length 1, giving a score of 7 (see Figure 6.10). This time it correlated with another *game equipment*, the *ball*. The definition of goal is *game equipment consisting of the place toward which players of a game try to advance a ball or puck in order to score points*. This is interesting because the same thing can be said about the *hole* in golf generally. The definition of *hole* in golf by this synset definition is *one playing period (from tee to green) on a golf course* which is about the round of play, not the physical hole.

It also suggested to move *terrains* from *golf* to *soccer*, with a depth of 7 minus branch length 1, giving a score of 6 (see Figure 6.8). Playing soccer on uneven terrain is also an interesting idea (see Figure 6.9) that should be possible to organize, although it may not be an evenly balanced game. One reason for why the interpretations of these suggestions worked well, may be

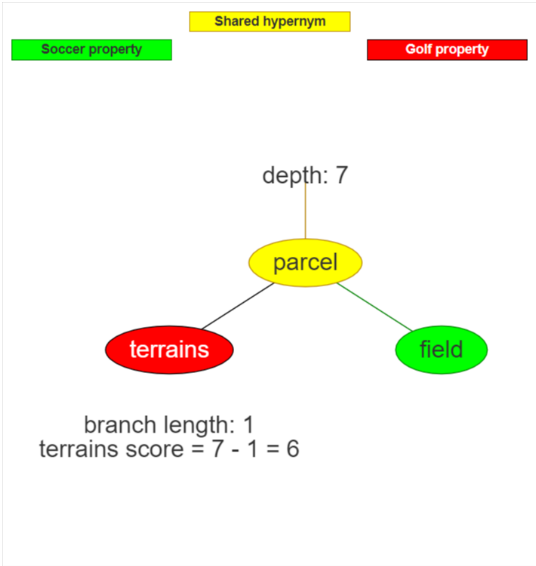

FIGURE 6.8: Top suggestion for moving something from golf to soccer. It suggests to move the property *terrains* (in red) from golf to soccer

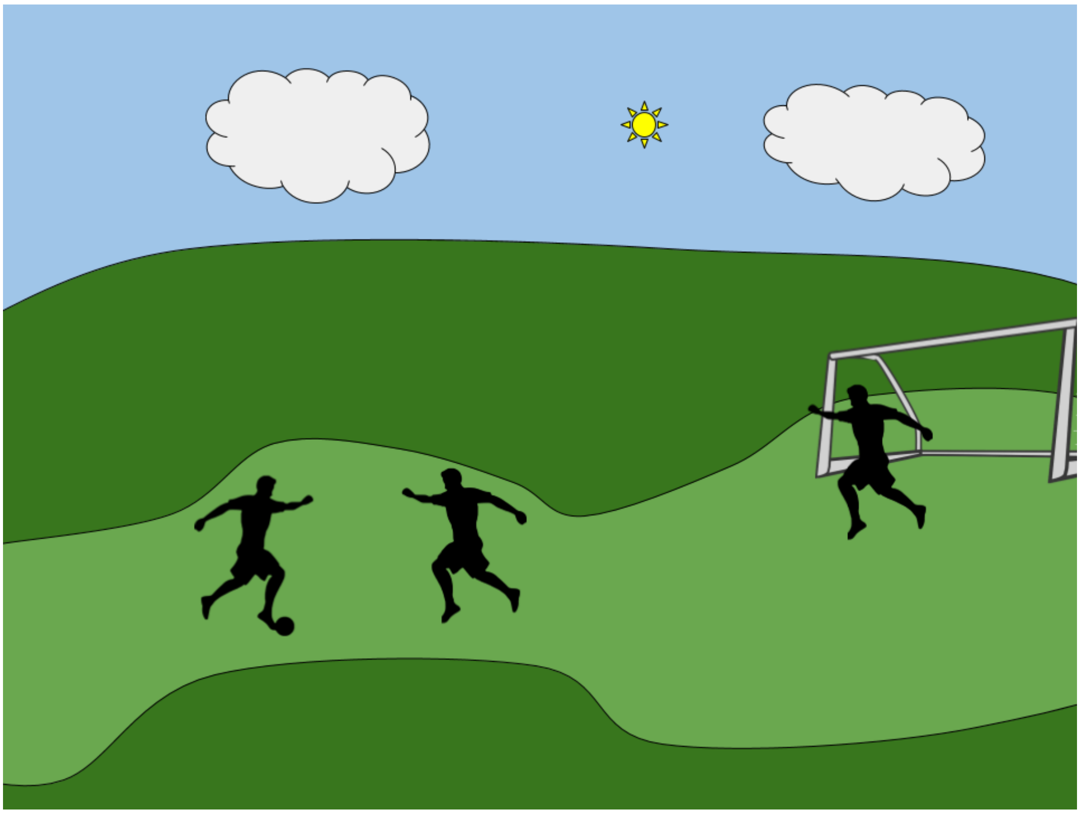

FIGURE 6.9: An interpretation of how the property *terrain* might be used in the game of soccer

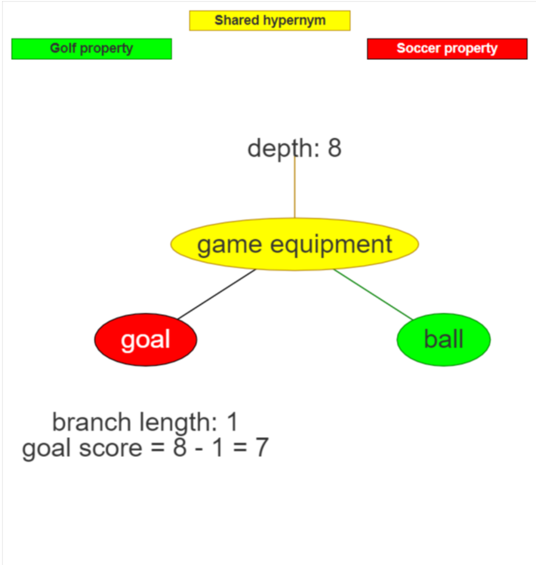

FIGURE 6.10: Top suggestion when blending soccer and golf. It suggests to move top property *goal* (in red) from soccer to golf.

because the relations to the other elements in the concept did not contradict. *Terrains* in golf matched with the part *field* in soccer through their common hypernym *parcel*. By not going further up the generalizing chain, the more likely the relations still work.

Chapter 7

Discussion

7.1 Aspects with approach/algorithm

7.1.1 Extracting information from Wikipedia abstracts

Retrieving the noun variant of a word

A common problem with the information extraction part of the application is that relevant properties are missing in the Wikipedia abstract. Sometimes they are there, but not in the form of nouns. For example cello has the adjective *bowed*, but it is far more relevant to find the noun *bow*, which is not found. A solution could be to find nouns by converting words from other part-of-speech to noun. WordNet seems to have the capability of returning some derivationally related forms of a word, but this was not included in the JavaScript library we used. Finding these computationally is not a straightforward process in itself, but could be done using a heuristic method where you remove or change suffixes of the word. With the word *bowed* you can search for the suffix *-ed* and remove this, but this method also turns words like *tuned* into totally different words like *tun*, which is a physical object we do not want to correlate with others. We would just discard any candidate word that does not return synsets in WordNet.

Sentence quality

One way to improve the information extraction may be by measuring the relevance of each sentence in the context of blending musical instruments. Some sentences contain cultural descriptions of instruments, rather than physical descriptions. These include geographical words like countries and people, styles, years etc... These are often collocating with *proper nouns*, used when describing individuals and places. Proper nouns are easy to detect since the initial letter is capitalized. Other characteristics of such sentences are numbers describing dates and years. Some of them are easy to identify by searching for patterns matching the conventions of writing them. For example, decades like 1580s can be found by searching for four numbers followed by an *s*, and centuries can be found by searching for one or two numbers followed by *th century*.

Sometimes an article try to explain a concept by describing what it is not. For example, if it includes phrases like *rather than* or *instead of*, it may suggest that the words afterwards are properties not typical of the original concept. We would rather select from the properties of sentences that describe very exact how the concept is, and not pick up properties that are known to be conflicting. By focusing the strategy on identifying words that are characteristic of deviations from the instrument, it could be possible to reduce unwanted properties by removing the sentences they came from.

7.1.2 Word sense disambiguation

Multiple relevant synsets

Some words have multiple synsets that are useful. For example the word *brass* in *brass instrument* has a synset for being a part of the brass instrument family, and also for being a material. Although the brass family synset is the correct one and also the one chosen in our program, the material is actually more useful when blending a brass instrument. The material is interesting because it is a part of the physical build of the instrument and something that can be transferred to other instruments. This move changes the other instruments sound and is therefore a more interesting creative choice, whereas brass family is just a categorization that is not really useful. One way to solve this could be to use all synsets that has a score over a certain threshold, as individual properties.

Duplicate words, multiple meanings

Abstracts often mention certain words multiple times. Usually they are meaning the same thing and are redundant. When using the *Simple Lesk* algorithm for WSD, the same synset is selected every time anyway. A problem with this is that a word may cancel out other important variants. For example, if the abstract contain *in the game of football* and *played by kicking a football*, the first football may cancel out the meaning of the last. One possibility of overcoming this problem is to narrow the context window to a part of the text, rather than the whole text. However this may remove important words from the context and reduce the general precision of the WSD.

7.1.3 Modelling relations in the data structure

The data structure we have used for the concepts in our solution is a list of synsets retrieved from words in a Wikipedia article. These synsets are intended to function as *being a part* of the concept. However sometimes the word has actually a different relation to the concept than being a part of it. For example, a guitar *pick's* main function is not to be a part of the guitar, but to *pluck* the strings.

Following this observation, it would make sense to use a data structure of a list of triples where the first item is the property, the second is the relation, and the third is the object the relation applies to. Blending similar relations might give more compatible suggestions. We then have a new problem in the information extraction step, to extract relations from descriptions like a Wikipedia article.

A method for extracting relations from text is using machine-reading systems like TEXTRUNNER (Banko et al., 2007). TEXTRUNNER can read enormous amounts of text and extract facts in the form of relational tuples. Approximately 80.4% of the facts were reviewed to be correct when it extracted 7.8 million tuples from the web. If we use a database of triples generated by a method such as TEXTRUNNER, we could filter it with pairs of properties in our text to search for common relations between the two. Then we could use the facts we found to see if they form a similar relation in the text. For example, if we query using the words *finger* and *string* from the Wikipedia abstract, *TextRunner* may return common relations such as *pluck*, *touch* and *strike*. If the word *pluck* is located nearby in the abstract, we can form a triple (*finger*, *pluck*, *string*).

Once we have formed triples, instead of looking for correlating properties, we could look for correlating triplets in the other instrument. We may find triples where item 1, 2 or 3 share a category with item 1, 2 or 3 of the correlating triple. Using the plucking triple we just created, we could find correlations with a triple such as (*hand*, *hit*, *string*). *Finger* and *hand* would share the hypernym *body part*, *pluck* and *hit* would share the verb hypernym *move*. The scoring of blending suggestions could then be based on a combination of generalization-based scores for each item in the triple.

7.1.4 Criteria

Branch length

In some instances, a shorter branch length score may not be better. For example if the abstract has both a word such as *butterfly valve* and its hypernym *valve*, the more general valve will get higher score and suggested as a better blend. Both words were originally referring to butterfly valve. On the other hand, a more general suggestion leads to more room for interpretation, which could lead to other implicit specializations since there are multiple types of valves.

Penalized categories

Which categories are the most useful for concept blending of musical instruments? We penalized *instruments* because we did not want to blend whole instruments into other instruments. *Individuals* were penalized because we did not want to consider people as a part of the instruments. Often these people were the inventor or famous musicians. We penalized *abstract entities* because it makes most sense to blend physical parts of the instruments. However categories like *shapes* and *materials* are also *abstract entities*, and these would be interesting to blend. Therefore it might be more useful to select the categories to include rather than penalize or exclude. The best method may be to use both. For example we include *physical entities* and *attributes*, but penalize/exclude *instruments* and *individuals*. Some categories such as *amounts* may have interesting information, but only if we can find and represent the original relation it had to an object in the data structure. For example the word *five* is not interesting to blend its own, but *five strings* is.

7.1.5 Other creative strategies

There are other relations possible to retrieve using WordNet that may be useful in blending strategies. WordNet link synsets to their *antonyms* which is their opposite meaning. The antonym of a top blending suggestion could be just as compatible creatively when blending. For instance, the antonym of *bowed*, *plucked*, would be useful in similar contexts as *bowed* itself. *Part meronyms* are parts of the corresponding word, which may be useful if they are of a different category than the word itself. Then they may form new connections in the generic space, leading to more interesting blending suggestions. We could go even further and find antonyms of meronyms as well.

7.1.6 Elaboration and interpretation

Some kinds of blending suggestions require further elaboration to be able to work in practice. For example, if the suggestion is *tone* as in the distinct tone of the original instrument, some elaboration has to be done in order for the instrument to generate the new tone in practice. Perhaps this distinct tone relies on other properties which did not join the blend. Making a cello sound like an oboe, is not obvious on its own. Tonal properties describing range like *baritone* and *soprano* are easier to interpret, since there are often existing variants of the instruments in different ranges. Such as the ranges of *cello* and *violin*. Other qualities like *bright* and *warm* may only need a small adjustment in the physical measurements to be realized.

Computational elaboration

How can elaboration be done computationally? This is an extensive problem, but we will list some techniques that may be helpful to improve the final output of the algorithm. Sometimes a new blended property conflicts with an existing one. One way to resolve this is to search for the antonyms of this word, and remove those. For example, if we blend the word *tall*, we can remove the antonym *short*. This technique may work best if relations have been accounted for, so that we do not remove *short* if it is describing a different part than the one we want to heighten. A similar technique may be to find an existing relation between the part and the object it is relating to. If we can replace this, we may resolve conflicts that are not antonyms. For example, if we blend *four* as in four strings, when the instrument already has six strings, we could replace *six* with *four*. This is assuming we have managed to model the quantification of *strings* as a relation, and preferably also a relation between the strings and the part of instrument that hold the strings. An instrument may have multiple sets of strings, such as drone strings, so it would be better to consider which set of strings the quantity is describing.

7.1.7 Goals

Conceptual blending can be used for open or divergent problem areas. The point would then be to create something new and novel, but still compatible and working. The interesting factor is the open and surprising creative choices that still work when the blended properties fit in the original system of relations. But conceptual blending can also be used in more converging problems, where the blending process is driven by a goal. The more specific the concepts you are looking for are, the more you can prune the search space to find exactly these type of results. In this thesis we have tried to use conceptual blending to invent new musical instruments. Before we pruned our search space using penalty categories, the results were very open and less related to our goal. The more carefully we pruned bad categories, the better blending results we got. Therefore it would seem that the focus of a goal-driven conceptual blending approach would be how to represent the goal and its ability to prune the search space in the right direction.

7.2 Application of algorithm

7.2.1 Human interpretation of blending results

The output of the application usually requires some interpretation by the user. Looking at the sentences where the original properties came from, may guide them on the way of understanding the possible relations involved. Even though the blending results may not give the ultimate blended version of two concepts, the results can be a spark for inspiration that the user may elaborate further. Blends made using wrong synsets, or forming bad correlations, may still make sense in one or another way. Sometimes the most creative suggestions are not the ones that are most compatible, but the ones that are surprising and radical. A creative human being using the program may interpret the results far beyond the intended meaning.

7.2.2 Different uses of the concept blending implementation

We have demonstrated that it is possible to make new instruments or some kind of extension of the existing one, although the application may need som manual aid for it to work optimally. One use case can be musicians that wants to explore new uses of their instrument. They can select their own instrument and a different instrument of choice for blending and the program will provide suggestions of properties to take from the other instrument and incorporate into their own instrument. Common examples of using elements from other instruments include the use of a violin bow on plucked instruments and cymbals or using a plectrum or percussive implements on bowed instruments. The use of conceptual blending on the domain of sports seemed to work well. The reason could be that properties of sports are more often related to the sport itself with a *part of* relation, than other more complex domains. The large number of sport terminology (especially golf-related) in WordNet is also success factor when blending sports.

7.3 Summary

The results of our algorithm rarely give us interesting correlations due to lacking a method of incorporating relations between the property and the concept or another property. We often get nice results that are possible to blend, but the correlations are mostly being part of a general categories like *device*, *implement*, *instrumentality* or *artifact*. The results are also limited by a lack of more specific niche words in the WordNet database, preventing several interesting correlations.

Chapter 8

Conclusion and future work

8.1 Future work

We believe an improved algorithm could be made by representing relations found in the abstracts in the data structure. The relations can be found using machine-reading systems like TEXTRUNNER. We can then generalize not only one word, but each element in the tuple. Research should be done on identifying the characteristics of the relations of good blending suggestions in order to rank the new results.

A good representation of relations may enable better techniques for resolving conflicts in the final blend. Conflicts may be found by looking for already existing relations similar to the one in the blending suggestion. The existing property may then be replaced or modified by the new one, rather than adding a separate conflicting property. Further research is needed to uncover the feasibility of these techniques.

8.2 Conclusion

Wikipedia abstracts are of varying quality for describing musical instruments. Removing cultural sentences improves the percentage of valuable properties, and carefully selected sentences with physical descriptions are needed to create a good representation of the instrument.

We did find a couple of characteristics that could indicate good blending suggestions. More specific shared hypernyms were a good indication of blending quality. But once the *depth value* went a certain length, it started to matter less. The difference between specific and even more specific was negligible because of the wide branching of categories. *Branch length* could also separate some good blending suggestions from bad, but not with the same influence as the depth value. We did not observe that the number of correlating properties had any specific influence on measuring the blending quality.

For specific blending domains, there are certain hypernym categories that indicate good or bad blending quality. These can be used to prune the search space. When blending musical instruments, we want to avoid properties that are individuals, body parts and most abstract entities except attributes like shape and material.

Our algorithm works best when the abstracts that are used are describing the physical parts of the instruments, since only a generic relation where the property is *being a part of* the concept is represented in the data structure. The suggestions could be more creative if the original relations were included in the generalization process as well.

Still the implementation bring creative suggestions ignites a spark of inspiration, motivating further elaboration by the user.

Bibliography

Banko, Michele et al. (2007). "Open Information Extraction from the Web." In: *IJCAI*. Vol. 7, pp. 2670–2676.

Benson, J. (2009). *Building Your First Personal Kanban*. URL: `http://www.personalkanban.com/pk/primers/building-your-first-personal-kanban/`.

Besold, Tarek Richard and Enric Plaza (2015). "Generalize and Blend: Concept Blending Based on Generalization, Analogy, and Amalgams." In: *ICCC*, pp. 150–157.

Boden, Margaret A (2004). *The creative mind: Myths and mechanisms*. Psychology Press.

Burckhardt, Philipp (2016). *wordnet-magic*. `https://github.com/Planeshifter/node-wordnet-magic`.

Busse, Thomas V and Richard S Mansfield (1980). "Theories of the creative process: A review and a perspective". In: *The Journal of Creative Behavior* 14.2, pp. 91–132.

Coulson, Seana (2001). *Semantic leaps: Frame-shifting and conceptual blending in meaning construction*. Cambridge University Press.

Fauconnier, Gilles and Mark Turner (1998a). "Conceptual integration networks". In: *Cognitive science* 22.2, pp. 133–187.

— (1998b). *Figure 6. Reprinted from "Conceptual integration networks"*.

— (2002). *The way we think: Conceptual blending and the mind's hidden complexities*. Basic Books, p. 37.

Fellbaum, Christiane (1998). *WordNet*. Wiley Online Library.

Kilgarriff, Adam and Joseph Rosenzweig (2000). "English Senseval: Report and Results." In: *LREC*. Vol. 6, p. 2.

Kim, Taeho (2014). *node-stanford-simple-nlp*. `https://github.com/xissy/node-stanford-simple-nlp`.

Koestler, Arthur (1964). *The act of creation*. London Hutchinson.

Lesk, Michael (1986). "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone". In: *Proceedings of the 5th annual international conference on Systems documentation*. ACM, pp. 24–26.

Li, Boyang et al. (2012). "Goal-driven conceptual blending: A computational approach for creativity". In: *Proceedings of the 2012 International Conference on Computational Creativity, Dublin, Ireland*, pp. 3–16.

Manning, Christopher D. et al. (2014). "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Association for Computational Linguistics (ACL) System Demonstrations*, pp. 55–60. URL: `http://www.aclweb.org/anthology/P/P14/P14-5010`.

March, Salvatore T and Gerald F Smith (1995a). "Design and natural science research on information technology". In: *Decision support systems* 15.4, pp. 251–266.

— (1995b). *Figure 1. A research framework. Adapted from "Design and natural science research on information technology"*.

Marcus, Mitchell P, Mary Ann Marcinkiewicz, and Beatrice Santorini (1993). "Building a large annotated corpus of English: The Penn Treebank". In: *Computational linguistics* 19.2, pp. 313–330.

Martinez, Maricarmen et al. (2011). "Towards a Domain-Independent Computational Framework for Theory Blending." In: *AAAI Fall Symposium: Advances in Cognitive Systems*.

Navigli, Roberto (2009). "Word sense disambiguation: A survey". In: *ACM Computing Surveys (CSUR)* 41.2, p. 10.

Ontañón, Santiago and Enric Plaza (2010). "Amalgams: A Formal Approach for Combining Multiple Case Solutions." In: *ICCBR*. Springer, pp. 257–271.

Popova, Maria (2012). *how intuition and the imagination fuel "rational" scientific discovery and creativity: a 1957 guide*. `https://www.brainpickings.org/2012/06/01/the-art-of-scientific-investigation-beveridge-2/`. Blog. Accessed: 2017-06-01.

Porter, Martin F (1980). "An algorithm for suffix stripping". In: *Program* 14.3, pp. 130–137.

Simon, Herbert A (1996). *The sciences of the artificial*. MIT press.

Turner, Mark (2014). *The origin of ideas: Blending, creativity, and the human spark*. Oxford University Press.

University, Princeton. *WordNet Statistics*. `http://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html`. Accessed: 2017-11-14.

Wells, Don (1999). *Daily Stand Up Meeting*. Accessed: 2017-11-27. URL: `http://www.extremeprogramming.org/rules/standupmeeting.html`.

Wikipedia (2017). *Statistics - Wikipedia, The Free Encyclopedia*. Accessed: 2017-11-24. URL: https://en.wikipedia.org/wiki/Special:Statistics.

Wormer, Titus (2017). *stemmer*. https://github.com/words/stemmer.