[SQUEAKING]

[RUSTLING]

[CLICKING]

**KIKI GUTIERREZ:** My name is Kiki. I'm visiting a scholar here at MIT. I'm actually an assistant professor at Polytechnical University of Madrid. My background is actually on engineering. I am aerospace engineer. I did my PhD on numerical methods, but after my postdoc, I decided to start doing things that really fascinates me, and that's how I end up doing something related with music.

So I'm relatively new in this field. Today I'm going to show you an algorithm that I came up for pattern detection in music, which is what I've been working on for the last year and a half, more or less. So as running example, I'm going to use the most listened song of the history, which is *Baby Shark.* So I don't know if I can play it. At least, it's the most listened song of the history of YouTube.

[MUSIC PLAYING]

Baby shark doo doo doo doo doo doo. Baby Shark doo doo doo doo doo doo. Baby Shark doo doo doo doo doo doo. Baby shark. Mommy shark doo doo doo doo doo. Mommy shark doo doo doo doo. Mommy shark doo doo doo doo. Mommy shark--

OK, is pretty much like that of the song. This is the melody. So what's the problem of pattern detection in music? We would like to have a tool, an automated tool that helps us to identify over the score or over a corpus the important musical ideas there, you know. So I'm highlighting here some fragments, how I look first at the blue fragment over there.

There are three fragments, three occurrences of them. All of them are exactly the same. They have the same notes with the same pitches and same durations. The only difference is that they occur at different moments of the score, but it seems logical. It seems reasonable to group them as the same musical idea and say that they represent the same pattern. We can call them-- we can put them a label, blue pattern.

For the green one, it's a little trickier because the occurrence number 2 and the occurrence number 3 are exactly the same. Probably makes sense to group them together. Occurrence number 1 has different durations for the notes and the occurrence number 4 is even trickier because they just start with the rest that-- it's three notes also and the lyrics. But it's pretty different.

So that's one of the main difficulties of the problem of pattern mining in music. That even though objects that we want to group together-- because in our mind represent the same musical idea-- over the table, they might look pretty different.

And there are two mechanisms involved here. The first one is something that we have already studied, transformations. So if we have a music fragment, if we apply a mathematical transformation over the notes there, we obtain other fragments that under some circumstances, our mind will process them as equivalent.

And how to deal with this computationally from the point of view of pattern mining algorithms? My approach was to use the concept of viewpoint. I'm going to show you an example of what is this about, but for the moment, it's enough to know that it has a double task.

On the first hand, it directly takes into account these transformations. You will see it in a minute. And probably more important, it simplifies the representation, because one of the constants in this course, in these classes, is that music is something complex. It has complex, logical structures among the elements of the score. And if we find a way to simplify that representation, would be helpful from the computational point of view.

So the viewpoint representation is pretty simple. It's just I'm going to substitute a complex tune like this one by a single sequence of symbols. So actually, here I'm showing three viewpoints representation. The pitch viewpoint, it's the sequence of these symbols 83, 83, 83, 85. It's the midi Note. Then we have the duration and the onset time.

I'm going to highlight a fragment. So you can see the different viewpoints representations for that fragment. And it's nice that the idea of constructing the score by overlapping layers of information. That's the idea.

Formerly, a viewpoint is a mapping between the current event, the current time slice, and all the former ones into symbol, usually an integer or float but could be others. Those that are derived at viewpoints. And perhaps for your own particular music application, you need to develop your own viewpoint representation.

And here is clearly highlighted how the transformations could be directly taken into account by using this technique. For instance, if you apply the octave transformation to that pattern there, that fragment, you obtain that one over there, which by the interval viewpoint representation, we have the same symbol. So for sure, any pattern mining algorithm that is trying to find this sequence will see that that one is the same.

Is that enough to deal with the concept of dissimilarity in music? Unfortunately, no. So have a look at these two occurrences of the green pattern of *Baby Shark.* There is no viewpoint representation that can take us from one to the other.

So someone said at some point, OK, it would be nice to have a kind of measure, a mathematical tool that help us to evaluate how different two fragments are. Like the aim here is that, OK, if I use that matrix with this guy and this guy, I would expect a lower value than the one that I would obtain when taking this guy and this guy. Because they are more far. They represent further ideas.

So in computer science, they use what is called the distance functions. This in a general scenario is that you can give to a function two objects, two random objects, for instance, me, myself, and this computer, and it returns a non-negative real number, evaluating the degree of dissimilarity between the two objects. In the context of sequences, because we are using viewpoint representation, so now this fragment will be a sequence. For instance. Let's think that I'm using the duration viewpoint. So in the literature, we have plenty of distance functions.

This was a very trending topic in music information retrieval 15 years ago, more or less. There were many studies and papers trying to figure out which was the best distance function, depending on the style. The Levenshtein distance function, perhaps, it's not the most advanced one, but this is one of the simplest one and probably the most used one. And it accounts for the number of substitutions, deletions, or insertions to go from one sequence to the other.

So to go from the sequence 0.5, 0.5, 0.5 to the sequence 0.5, 0.52, we just need a single substitution. So the Levenshtein distance function between the fragment 2 and fragment 4 is 1. Probably the Levenshtein distance function between this guy and this guy it's-- I don't know-- two or three, at least. So cool.

Now we have all the ingredients needed to understand the inputs and outputs of my algorithm. We have a corpus or a piece of music. We have also the viewpoint representation that the user wants to choose. And some parameters at the input. The minimum support is the minimum number of repetitions that we require to a pattern to be considered frequent and to be included in the result list.

These two guys, the minimum length and maximum length, are the parameters to control the lengths of the patterns that we want to mine and some parameters to control the degree of dissimilarity from the former slide. And the result is just like the list of frequent patterns together with the position. This is an example with the piece of the former slide. Now I'm selecting the interval viewpoint and those parameters over there. And this is an example of the output that we might obtain.

So we have a yellow pattern that has four occurrences and have a look that each of these fragments are at most distance one from any of the rest, taking the Levenshtein distance function. And we might have, for instance, also this guy, the purple pattern, and this green one, which is of length 4. So in a real case, the pattern structure might be complex, as you see, because you have nested patterns. You have many overlappings.

I won't go into the details of how I implemented the algorithm, just a brief overview. I divided in four steps. The one is the creation of an initial database. A vertical database is called sometimes, where I just take a sliding window of minimum length, the minimum length selected by the user, and I just create a database of all the patterns together with their positions within the score.

Then in the next step, I compare every pattern against all the rest, trying to see if they satisfies the distance constraint. And in that case, I group them together into what I call metric patterns, which are like groups of fragments of music. At this stage, it's safe to delete the entries that doesn't reach the minimum support constraint. They don't have enough occurrences to be considered frequent.

And finally, as here, I still have patterns of length equal to the minimum length, and we actually want longer patterns. So the last stitch is overlapping different patterns to form longer ones.

And to finish this I'm going to show you some results. This is the last slide. I'm actually already using this. I started collaborating with these research groups like seven months ago. This is a research group from another university of Madrid. They got a good amount of money of the European Union to study the Italian operas.

They have a huge database with many features for each piece. And in particular, they are interested in the motion that the piece evokes. So here, we are trying to find some correlations between the emotions and the pattern structure of the different pieces.

These two other products are pretty similar. The number 2 is on a corpus of folk music from a very concrete region of Europe, around the Pyrenees, and the second one is the database of jazz solos. So here we are building classifiers, similar to what we did in the last class.

Here I'm showing the number of patterns per style for the different styles in this corpus, but perhaps here, we can see a cool thing of the algorithm. That it returns also the position of the patterns, not only the fact that a piece contains the pattern.

So here, I'm plotting-- it's called coverage, this variable, but it's actually like the probability function of encountering a pattern within a solo. So I normalize the length of all the solos by style, and the thin line is the average for the solos.

We can see, for instance, that the concentration of patterns tends to be at the beginning and not in the end. This is intuitive because the improvisers usually start their solos in a more organized way, taking thematic material from the original melody, and then they start to do more crazy stuff.

Then, I'm also trying to build up a pattern dictionary in an Irish corpus. These are the five most frequent patterns in the subset of minor rules in the corpus that I'm working on. So I don't know for a performance that is interested in playing this music. Perhaps, the take away of this is, OK, you might want to start studying these patterns because they appear quite often in the corpus. So better to first master them.

For the future with Michael, we want to study how the algorithm performs in a nice corpus of early music that he has, and also, we want to analyze some solos of Charlie Parker, who was one of the most mathematical improvisers of the history. And also for the close future, I would like to see if this tool can be helpful for plagiarism detection. I suspect that two pieces that are very similar are also some similarities in their pattern structure. And that's all. Thank you very much guys. Any questions.

[APPLAUSE]

**MICHAEL CUTHBERT:** By the way, just so you get a sense for your final things, that was 12 minutes. So a tiny bit longer than what you have as a maximum.

**KIKI GUTIERREZ:** Cool. If any of you are interested in a particular thing of the algorithm, just drop me an email, and we can talk about it. Thank you.

**MICHAEL CUTHBERT:** So what I want to do is I want to immediately start by putting some of these thoughts to work. So get with a partner next to you. You can do that. Over here, we need a group of three. Or unless you want to participate with a friend or jump over there or jump over with Jordan, if you do.

And here are six melodies. I'm going to play each one of them. I'm going to play A bunch of times also. So I'll tell you which one I'm playing, and you're going to-- and you can talk during this or wait till just after playing. You're going to tell me which melody is most or least similar to A.

[PLAYING INSTRUMENT]

OK. And this is B. And I want you to say why.

[PLAYING INSTRUMENT]

This is C.

[PLAYING INSTRUMENT]

And I'm going to play A one more time to get it back into our heads.

[PLAYING INSTRUMENT]

And now D-- Oh, and by the way, this isn't a right-- isn't a right answer one.

[PLAYING INSTRUMENT]

Great. Now E.

[PLAYING INSTRUMENT]

And now A one more time before doing F. So this is A again.

[PLAYING INSTRUMENT]

And now F.

[PLAYING INSTRUMENT]

Oops, that's F and E simultaneously, which is very different. Now F.

[PLAYING INSTRUMENT]

Great. Talk amongst yourself. I want a ranking from your group, so somebody write it down. And I want to know why. OK, let's bring it all back together. Let's bring it all back together. So look looking at your list, and we'll count like normal human beings. Whichever one is the top in your list is one. Whichever one second, two, stuff. And we'll vote by fingers first off for the general ranking. So if you hold up the number of fingers that-- if it's closest, it's pulled up one finger. If not, two-- whatever. And full palm means I have no idea what that is.

OK, B. Oh, B's doing pretty well A lot of twos, a couple of ones. C. OK, well, a little bit more variety. Hold them up so that other people can see also. Look around the room. It's not just for me. Great. Great. D. Oh, OK, we got a variety who don't know, whatever. Great. E. Oh, still see some poems or things. Great. And F. OK. Oh, we have one four. Good, good. What do you-- you put it down. What do you guys-- those who put F as four, what do you put as five?

**AUDIENCE:** D.

**MICHAEL CUTHBERT:** B.

**AUDIENCE:** D.

**MICHAEL CUTHBERT:** D. OK, D. Good. Good. Great. Somebody who had C above B justify your answer. Who had C above B? There were a couple people. Yeah, go ahead. Jake first. Yeah, groups.

**AUDIENCE:** C was basically just a transposition, whereas B like-- B changed a lot of the rhythms a bit. B, I think, fits the exact pitches a little better, but-- or aside from a couple places where it has some accidentals, but it changes the rhythm quite a bit. Whereas C is just a transposition. So for relative pitch essentially. We waited transposition equivalence more.

**MICHAEL CUTHBERT:** OK, good. Good. I'm already hearing words that I'm liking stuff. Great. Somebody who had it the other way around justify your answer. Who had it the other way around? Who had-- a bunch of people had B above C. Yeah, John.

**AUDIENCE:** I'm looking for the sound feel that A had. Because a lot of the comments are probably similar to A and B. That's why we put it a bit higher relative to C. And when you take transposition into account, only part of C's becoming transpose, so it doesn't even feel like it's like a full transposition.

**MICHAEL CUTHBERT:** Great. Only part of it's transpose. Yeah, yeah, it kind of gets back on for a bit and then comes back off. Great. Who had D-- who had D one two? Anybody? I can't remember. Why do you have one or two?

**AUDIENCE:** Because it basically has this-- so it has all of the same notes in the right positions-- or basically-- yeah, all you have to do is remove notes, and then you get the same thing, and-- I think with a few exceptions. But that never happened in any of them.

**MICHAEL CUTHBERT:** Great. Who had D very low? Yeah, go ahead, Tony.

**AUDIENCE:** Like before--

**MICHAEL CUTHBERT:** Sorry, Tony. Can you speak louder?

**AUDIENCE:** Before, like a rhythm, I guess, the same.

**MICHAEL CUTHBERT:** OK, great. Did anybody have E above four? OK.

**AUDIENCE:** We have D at three. And we put it there because the pattern was very similar, if not identical, even though the melody wasn't all that close. So we figured that probably counted for something.

**MICHAEL CUTHBERT:** So when you say pattern, what's--

**AUDIENCE:** It's like the rhythm.

**MICHAEL CUTHBERT:** The rhythm. The rhythmic pattern, it's about the same. Good.

**AUDIENCE:** That's an inversion, right?

**MICHAEL CUTHBERT:** Is it an inversion? No. No. Would I do that?

**AUDIENCE:** The inversion.

**MICHAEL CUTHBERT:** By the way, I can't remember where melody-- I think melody A comes from a Huron book and then gives-- there's some search book, and I should have my notes better, and I'll try to make sure it gets annotated later. That had three other melodies. I tried this in the past, and it was so obvious that everybody had the exact same ranking. I had to agree with them. But I think this is better for making some arguments. Good. Who had F anything but-- actually, somebody who gave F five? Jonathan, why would you-- did you give F five?

**AUDIENCE:** Yeah.

**MICHAEL CUTHBERT:** Why did you give it five?

**AUDIENCE:** I mean, it didn't have any noticeable similarities. Like at first, it seemed closer to E in terms of-- it might have been inversion but then not really. The rhythm is also completely different-- or not completely, but it's fairly different.

**MICHAEL CUTHBERT:** Great. So I'm just going to point, and we'll get some-- just what your ranking is. So say them from-- Matthew, what was yours?

**AUDIENCE:** Alphabetical order.

**MICHAEL CUTHBERT:** B, C, D, E, F-- OK, good.

**AUDIENCE:** B, C, D, E, F.

**MICHAEL CUTHBERT:** B, C, D, E, F.

**AUDIENCE:** [INAUDIBLE]

**MICHAEL CUTHBERT:** B, C, E, D, F. Great. Great. Vincent?

**AUDIENCE:** C, B, D--

**MICHAEL CUTHBERT:** C, B, D-- good. And anything it feels like you're not being represented on there? Hannah, what's your group have?

**AUDIENCE:** I think we put C, B, D, E, F.

**MICHAEL CUTHBERT:** C, B, D, E, F-- great, super. Now what I want you to do-- we're not going to get through all of the exercises today, but I think this is the most important part. What I want you to do is think about what ways-- I'll give you a little bit of things-- what are some ways you can make sure that your computer system that is going to classify things by similarity follows your intuition of what is similar, and not somebody else's intuition for what is similar? So that's going to be the main theme for the rest of this.

So we are intelligent people. We are intelligent musicians. We make these choices, and yet we are making differences on how far and how similar they are. So, in fact, I'm going to blank the screen and say the one takeaway from today's lecture, I hope, and from all these things, is that there is no right answer for the similarity between two melodies, between the similarity between two pieces.

There may be wrong answers. I will not deny that, that if somebody said that F was closer to, I don't know, than the same thing with one note changed or something, I would think that that might be wrong, that your program might be malfunctioning.

But there isn't a right answer. And a lot of it-- what's different between good answers are what we think of as important when thinking similarity. There is a yearly competition-- I think it's been suspended since COVID, so I don't know if it's restarted, but for the algorithm that can classify songs as the most similar. And here is a place where I would say, what are your ground truths? How do we trust that you have gotten it right? And are we just trying having to recreate the views-- I won't say biases, but the views of the people who organize the conference and what's that going to do for us?

So I want you to start thinking that. And I will tell you what F is beforehand. F is one that a lot of computers' programs-- in fact, what I went aha during one of these algorithms, they could-- F is 1 that a number of algorithms, especially older ones, will classify as the most similar.

Because what is F? Unlike any of the other lines, F has every single note and every single rhythm, if I did it right. I was doing in my head. Every single note and every single rhythm from A-- just order didn't matter. A is the counterset function, the unordered version, the P-- yeah, the permutation does not matter version of F. Or F is the permutation does not matter version of A. Did I get it right?

**AUDIENCE:** It looks right.

**AUDIENCE:** It looks right.

**MICHAEL CUTHBERT:** So we're going to go quickly through some things I think you've probably seen before, some ways of measuring distance. You all learned this at some point, the Euclidean distance between two points. Take the square root of the x terms. Take the square root of the y term. Add, what, difference squared plus difference squared square root. Square root of x squared plus difference between x and difference between y.

So anyone seen this thing, where the distance between these two points, 3 comma 2 and 7 comma 8, is 10. Taxicab distance-- Manhattan distance, we'll go with taxicabs since not all of us have been to a Manhattan and had the joys of taking a taxi there. And so why this-- here, the distance was 10. Here, it's approximately-- that's not a negative sign. That's an approximate sign-- approximately 7.2.

Why is the distance greater here? Somebody who's done this triangle inequality. Talk English to me for a second. Talk like you're talking to your cab driver who you're explaining to this-- cab drivers are really smart. Talk, but who may not have heard the final inequality. What is represented by the term Manhattan distance or taxicab distance? What's the notion-- intuition? Yeah?

**AUDIENCE:** Go along the axes.

**MICHAEL CUTHBERT:** Go along the axes or that go along-- let's get more literal. One of the things that we don't do so well is step back into the real world. What is the distance traveled? What constrains the taxicab from not hitting distance of 7.2 but instead of 10? Yeah?

**AUDIENCE:** You get straight up and down to the side.

**MICHAEL CUTHBERT:** You can only go straight up and down to the side. You can only go on-- let's go even further back. What, in Manhattan, if you don't want to get arrested, you can only drive on?

**AUDIENCE:** Streets.

**MICHAEL CUTHBERT:** Streets. And the streets in Manhattan go--

**AUDIENCE:** Orthogonal.

**MICHAEL CUTHBERT:** Yeah, they're orthogonal. There are these little lines. So you are constrained in where you can go. So if there are constraints on your distance, and the most common one is you can go up, down, left, or right. You can't always do that in Manhattan but because of one ways. But let's assume that we have certain constraints. You can be brought down. Good. I wanted to make sure that we all have that, and so that we can start thinking about-- first off, that what operations are allowed determines the distance metric. What operations are allowed determines how far the distance are.

What are some operations we do in music? That's a question. What operations do we allow and not allow? Adam?

**AUDIENCE:** We could look at many different models.

**MICHAEL CUTHBERT:** We can look at midi difference between notes. So therefore, we can take notes and bring them higher and lower. We can raise and lower notes. What are other things we can do? What are some things you've ever done with a piece to make it a little bit different or interesting? Yeah?

**AUDIENCE:** You can subdivide or combine notes.

**MICHAEL CUTHBERT:** You can subdivide or combine notes. Maybe you can, maybe you can't. But yeah, quite often you can. This is a context where you can. Yeah, other--

**AUDIENCE:** --durations.

**MICHAEL CUTHBERT:** You can change durations-- great, super. How about this? Which of these two chords are closer to the first one? The first one is going to be C major versus-- another little similarity problem. The second one was G major. Sorry, the first one was G major. The second one, I went from C major to C augmented. Great, C augmented triad. So those are two things.

Which one? Who votes that from going from C major to G major is closer? Who votes that C major and C augmented are closer? Two people. OK, great. So a lot of it has to do with your thought about-- well, on the augmented, you're only changing one note, and you're only changing by a half step, the minimum distance in our Manhattanized musical world of midi and piano keyboards. That is, their minimum distance is one half step-- not for all music in the world. Great. C major to G major-- you're also just moving one. If you think of something this way, you're moving one distance in what space?

**AUDIENCE:** [INAUDIBLE]

**MICHAEL CUTHBERT:** Oh, what's that word?

**AUDIENCE:** Circle of fifths.

**MICHAEL CUTHBERT:** It's circle of fifths space. C and G are about as close as you can get without being the identity, C and F probably the other way, although, I don't know, maybe it's a one way circle of fifths. You only go around one direction or another. Great.

Based on the time, I'm not going to go through all these other measurements of distance that people can do. Who has heard of earthmover distance? That is the amount of work that it takes to move one mound of things over to another place. And sometimes, you're optimizing depending on how much it costs to move distance and how much it costs to move material. You can end up with different results.

This was one of the charts I think I showed early in the semester is here's one place where earthmover distance might be a good use of things of distances. And then Levenshtein or edit distance is what was mentioned in-- do you use it in your work? Yep, so that's where you're talking about, so the idea of how to change the word Hyundai into Honda, and no international East Asian politics please for a second. And you can think of every time, OK, H and H are the same. So it has a cost of 0. Or we can delete the H and start an O, and we have a cost of 1. But we can find the pattern of as we change, we're going to insert a Y after the H.

We're going to substitute a U for an O. This should be symmetrical the other way around-- different operations. N is the same, so that's good. D is the same, so it doesn't cost anything. So our cost function goes here. And so we're trying to find the minimum cost from going from one end to another. We don't have time to go through all of the algorithms for this. But Levenshtein distance, edit distance, has a lot of good qualities that makes it useful for a lot of musical similarity tasks. Just so that you can say your professor at least put the algorithm up on the hand for a second.

But more importantly, I think a lot of times is thinking about the particular costs of things in a musical space, in a musical world. So for instance, is deleting what-- we're trying to think about two pieces, two melodies. One of them deletes the first note. What would you call the cost on that? Well, maybe 1.

But then, if it doesn't make up the total rhythm later, and everything from here on is going to be off, and it's one line within an orchestral piece, that might be a higher cost, maybe some-- and the classic debate is whether changing a note is that the same or changing a letter in something like this? Is this the same? Does this cost 1 or does this cost-- well, one way you can change a letter is you delete it, and then you add a new letter back with a cost of 2. And these are things that come up quite a bit in similarity search.

And just really want to say that it comes up a lot in music-- don't borrow your distance metric from somebody else. Different ones might be used for different situations. So the distance between dog and gato-- well, we can substitute d for g. I don't know, or maybe we add other things. But I'm going to assert that, in some situations, the distance might be 2 between dog and gato. What we do is we use the substitute closely related pet function for cost of 1. So dog becomes cat, and then translate English to Spanish might cost 1.

And if you think about large language learning models and things, you might want to have functions like this. And, in fact, this is not a digital humanities text class. But if it were and we were doing computation, we'd definitely be talking about an algorithm called word2vec, which was one of the earlier successful algorithms for trying to predict what words are similar to other words, what words are synonyms, so you can create a kind of cost function that is for this word is a synonym for this one that has been substituted that is lower than this-- then this sentence is different from this one because it's using a completely different concept. I'll skip that.

So when we're thinking about these distances and these weird things like substitute dog for cat on low-cost substitute cat for gato at low cost, what's the term that we spent a lot of time, maybe even too much time for-- it felt like at the time-- talking about earlier in this semester, that helps to think about things that are not the same, but might be closely related to each other?

**AUDIENCE:** Equivalence.

**MICHAEL CUTHBERT:** Equivalence, or equivalence classes, yes. So one of the things you might want to do is define what equivalence classes it could be. I mean, I think last-- other times, I've given the exact same melody up an octave, and everybody immediately said, oh, that is basically the same thing. So everybody was very quickly putting in an oh, equivalence class things. So I wanted to make sure that we had that.

And so once you have these distances, we tend to go through-- and this is if you're in a biology class, you'll spend a lot of computational biology, a lot of time on this-- sequence alignment, a kind of distance metric where you're trying to find the minimum distance between two things that you believe might represent the same type of thing. Or you might say it's innocent until proven guilty. We'll first try to see if they can be changed into another thing at a low cost and then discard once we realize the cost cannot be minimized.

I will say that algorithms that can be short circuited, that you can prove at a certain point you can't do better than this cost will speed up a lot of your run times because once-- you might say that there's no way that this could be better than 20% or it could be-- yeah, there's no way that this could possibly be better than 90% similar to this. So I'm going to stop looking at the rest of the piece or whatever your cut off.

So one of the classic things for sequence alignment is trying to find-- this is Google's data set they released at the height of Britney Spears popularity of all the number of searches that they believed were trying to find the top left one, Britney Spears-- actually really, really impressed that the number of correct spellings of a hard name to spell outweighs the rest. Anyhow, that's all we're talking.

And the people who are really, really good at this-- and any time I'm trying to figure out a similarity sequence alignment or a similarity task that I don't know is to look at the people who are trying to align base pairs in biology or trying to align genes because they have many, many options.

So I'm just going to keep pounding this term in as many different ways I can do. All the things that we're just working with-- Hyundai, Honda, Britney Spears, genes, those are all strings. But we work on notes and clefs and things like notes and stuff, but things like that. So how do we get them in?

So this is great to get this from two different people, same thing-- we use things called hashes, which are very similar to the concept of viewpoints to the rescue, so that-- try to convert things-- hash and note. We might say that here are equivalence classes all notes that are names with octave. And so we might hash a stream by just joining all the hash notes for all the notes in there.

You will find, in Music 21, if you're working on it, there's a bunch of tools for this already. They're in music21.search, a module that we have not talked about now and we will not talk about again. But if you're doing a lot of searching, it's probably worth reading the module reference for it. I think that there might be a user's guide, but I can't remember if I finished it or if it just trails off after a few words.

So we might take a string, convert it to a stream-- that's hard to say very fast-- and translate it. And we might have some of hash function that tries to make everything into an ASCII character. Though there's no reason that everything needs to be turned into a string like nameWithOctave.

In a lot of ways, strings are just arrays of ints. We're talking about that A-- no, lowercase a is generally represented internally as anyone-- remember number?

**AUDIENCE:** 97.

**MICHAEL CUTHBERT:** What's that? 97 or 96, I can't remember. 96? Yep, and capital A-- is that one 60--

**AUDIENCE:** 65.

**MICHAEL CUTHBERT:** 65. OK, so some people some people know these. I used to have them all top of the head. So all the letters you're doing have a particular representation. And back in the bad, bad days of the '60s and '70s, different computers would have different representations for this, and then we all agreed on the same representation for letters. And then we remembered that there are other things in the-- other characters in the world. That looks too much like an A. How do I do a jin or something, or an alpha, beta, things like that. And then, for a while, we had a big problem that they weren't all converging to the same thing.

Anyhow, digression aside, maybe we'll get to the point where we can start converting things besides midi numbers and notes into something more standardized because, right now, the midi numbers is basically the only standardized notes, which is probably why midi keeps being used for a lot of computational projects.

So the hard part is always finding out what numbers we should use to represent a note. So if we're going to convert nameWithOctave and we want to make a string, and then we want to make it a number, and then we want to have a whole bunch of numbers, what have we just recently seen that looks like a tool to take a score or a part or something and works like a hash or a viewpoint that tries to convert it to a bunch of numbers? Not asking you to think too far back, but farther back than today. Yeah?

**AUDIENCE:** I'm getting a feature representation.

**MICHAEL CUTHBERT:** Yeah, extracting features, getting a feature representation. Yeah, so feature extraction and this kind of viewpoint searching go hand-in-hand with each other. So if it's partially why once you finish up a search function, you're just going to want to probably try to see if AI or machine learning can do it better because you have everything ready to go for it. But sometimes what we extract is different from others.

I want to give a little bit of a caution that back then-- of course, you had to do a little final, a final project called the UAP-- and we thought that making these viewpoints, making a hashing system for Music 21 for comparisons would be a nice senior project. And then we both realized that, no, it's a lot bigger than we thought and a lot more complex than we thought, and so it needed to be an M. Eng. Emily Zhang was great at creating this and great, oh, we did a really great M. Eng project.

And then we realized, no, this really needs to be a PhD project. We did not continue on-- there are so many difficult parts of hash algorithms because you want to think about things like-- yeah, we'll not get to it-- going all the way back to the beginning, how can we create a viewpoint or something that allows D not to be totally, totally different for anybody who didn't put D as the last of all possible results? What kinds of hashes-- what kinds of numbers would we need to represent a piece on to make D not the worst and really make sure that F isn't the best?

So that's going to be our last 5-6 minutes of class. I want you to talk with 5 minutes of y'all talking with each other, and 5 minutes of y'all talk and talking to me. So what kinds of feature extraction, what kind of hash function, what kind of viewpoints? These are all slightly different concepts, but they're all in the same area. What kinds of equivalence classes will you need in order to make this happen? Go ahead.

OK, I hear words continuing but less frequently. Let's talk about what are some of the ways that people thought to create a strategy that doesn't make D and F about the same? Yeah, go ahead.

**AUDIENCE:** You could look at the sequence of local maxima and minima.

**MICHAEL CUTHBERT:** Local maxima and minima. OK, I think I know what you're talking about, but let's give you a little example. Let's talk about A. What do you--

**AUDIENCE:** So you could maybe argue that the C is the local minima, and then the F is higher than both of its neighbors, so it's a maximum. And then a D is a minimum, the E's a maximum, and then the D and C after that are not really anything until you hit the A on the 16th note.

**MICHAEL CUTHBERT:** Cool. So yeah, we're just looking at every time the direction changes of the pitches. Great. And compare that-- so beginning A has G, F, D, E. Here, we have D, F, G-- a tiny bit different, D, F, but then going down to-- a little bit different, but it's at least giving some numbers we have. Always the question is, does your current streak end when you hit a rest or not? And maybe it depends on how long the rest is, so good. Other strategies? Adam?

**AUDIENCE:** I would look at where offsets are the same and then check that their notes are the same or not.

**MICHAEL CUTHBERT:** Great. So we're going to look at offsets that are the same and see if notes are the same or not. That works really well. And what I'd love to do, if this were, what do you call it, the generalized adversarial problem set, the gain problem set, where one team has to solve the problem. The other team has to keep giving them things that break that. I think it was a great idea. And I think it would work in general.

But I could generate something where all I insert is let's insert a 64th rest at the beginning and then put all random notes. And you're going to end up with-- and then maybe we'll put one note that's the same at the end. And you could end up with 100% of the notes on the same offset are the same. I think we would really work in the real world, but we might want to always think about something like that, too. Great idea. John? Then two people.

**AUDIENCE:** [INAUDIBLE] so first--

**MICHAEL CUTHBERT:** You can say--

**AUDIENCE:** --builds up on what Adam said

**AUDIENCE:** But first, you take a look at the notes and do a set, kind of like crossover between a set of D's and aces and then ASes. So both of those would still show up relatively high. And then compare the offsets, which would obviously take up a bit, but still keep D relatively high.

**MICHAEL CUTHBERT:** Super. When we're talking about offsets, are we talking about-- what kind of offsets?

**AUDIENCE:** We're in the-- I guess within the two measures that the notes being--

**MICHAEL CUTHBERT:** Great. Where in the two measures? Where in the measure? We sometimes want to do global offset from the beginning of the measure. But then you can't identify similar phrases or, all it takes is put a repeat, put the first four measures, repeat it once, and suddenly the whole rest of the piece is different. So yeah, that's great.

**AUDIENCE:** If you just start at time equals 0 and go all the way through the piece, like anytime the two pieces have the same pitch, you score-- so the alpha that they both have on beat two would be like a quarter point because--

**MICHAEL CUTHBERT:** The F that they both have on beat two would be a quarter point because--

**AUDIENCE:** Their duration is only for that 16th note.

**MICHAEL CUTHBERT:** Got you. So we look at shared duration. Great. I like that a lot. Did anybody try to come up with a equivalent? Yeah? The contour ends up being a kind of a new equivalence class that we hadn't talked about, which kind of works out in thinking that everything that doesn't change directions is a kind of passing tone, even though not in the proper music theory term, and so can be ignored.

By the way, the one I use quite often is I'm just going to look on downbeats or on beats and ignore everything else, and that works pretty well for a lot of things.